Реализация базовых операций целочисленной арифметики в гетерогенных системах*

А.В. Панюков, С.Ю. Лесовой

Национальный исследовательский университет ЮУрГУ

В работе рассмотрены вопросы масштабируемости выполнения операций целочисленной арифметики с операндами произвольной длины. Отмечено, что классический алгоритм деления «столбиком», в отличие от остальных, не является масштабируемым. Предложено решение проблемы масштабируемости операции деления посредством применения метода Ньютона. Даны оценки сложности и масштабируемости.

1. Введение

При программировании вычислительных задач, современные компьютеры позволяют использовать не только ресурсы центрального процессора, но и графического ядра, представляя собой гетерогенную систему с высоким вычислительным потенциалом. Воспользоваться этим потенциалом позволяет, например, высокоуровневая платформа OpenCL, предоставляющая средства управления ресурсами графического ядра в дополнение к центральному процессору.

Длинные числа являются попыткой преодолеть ограничение платформ на диапазон представления целых чисел, определяемый типами данных, поддерживаемых стандартами языка. Необходимость в очень больших целых числах возникает при использовании криптографических алгоритмов или при решении задач, требующих точных дробно-рациональных вычислений. Проблема увеличения производительности выполнения операций над длинными числами остается актуальным уже длительное время. Основные результаты алгоритмического подхода подробно представлены в классическом труде Д. Кнута [1]. С появлением многопроцессорных систем стало возможным дальнейшее повышение эффективности таких операций на основе параллельных версий классических алгоритмов.

Возможности различных параллельных реализаций операции умножения: алгоритм Шенхаге – Штрассена, алгоритм Карацубы, алгоритм Тоома –Кука, – рассмотрены, например, в статье Е.Г. Качко [2], где отмечена конкурентоспособность классического алгоритма умножения «столбиком» при большом числе параллельных процессов.

В работе [3] предложены реализации целочисленных операций с применением классических алгоритмов. В частности рассмотрены операции сложения–вычитания, умножения на цифру (под цифрой понимается минимальный неделимый элемент чисел произвольной длины) и умножение чисел произвольной длины. Отмечено, что классический алгоритм деления «столбиком», в отличие от остальных, не является масштабируемым, т.е. наличие многопроцессорной среды не повышает его производительность.

В данной работе, являющейся развитием работы [3], предложено решение проблемы масштабируемости операции деления посредством применения метода Ньютона. Даны оценки сложности и масштабируемости предложенных алгоритмов.

2. Теоретические оценки возможности реализации

Приведенные ниже оценки сложности выполнения операций алгоритмами, предложенными в работе [3], были сделаны при ряде условностей и допущений, а именно:

а) количество доступных вычислительных элементов GPU не менее чем разрядность меньшего из операндов в случае операции сложения, а в случае операции умножения – не менее произведения разрядности операндов.

* Работа поддержана РФФИ (проект 10-07-96003-р_урал_а)

б) объем доступной оперативной/регистровой памяти GPU позволяет разместить исходные числа, результат и все промежуточные значения.

в) не учитывается пересылка данных между оперативной памятью и памятью GPU, предполагается, что исходные данные уже находятся в видеопамяти и результаты остаются в ней же.

Приведенные допущения позволяют получить оценки времени выполнения основных арифметических операций, приведенные в табл. 1.

Таблица 1. Оценки среднего времени выполнения операций с помощью классических алгоритмов

Операция	Оценка времени выполнения	Количество параллельных процессов
Сложение	$2t_A$	$L + 1$
Умножение на цифру	t_M+2t_A	$U + 1$
Умножение	$t_M+2t_A + 2t_A\log_2 L$	$L + U + 1$
Деление на цифру	$t_D U$	1
Деление	$(U – L + 1)(t_M+4t_A)$	$L+1$

В табл. 1 приняты следующие обозначения:

1) t_A – время выполнения обычной операции сложения над разрядом длинного числа одним вычислительным элементом,
2) t_M - время выполнения обычной операции умножения над разрядом длинного числа одним вычислительным элементом,
3) t_D - время выполнения обычной операции деления над разрядом длинного числа одним вычислительным элементом,
4) L – длина наименьшего из операндов (количество значащих цифр в используемой позиционной системе счисления),
5) U – длина наибольшего из операндов.

Из таблицы 1 видно, что классический алгоритм деления «столбиком» на цифру, в отличие от остальных, не является масштабируемым (при увеличении числа доступных процессов, время выполнения не изменяется, дополнительные процессы не могут быть использованы), а время его выполнения линейно зависит от разрядности чисел. При делении на многоразрядное число время также линейно зависит от разрядности чисел, но появляется возможность использования дополнительных процессов.

В работе [1, с. 304 – 425] показан способ реализации в двоичной системе счисления операции деления через операцию умножения с помощью модифицированного алгоритма Ньютона. В следующем разделе приведена реализация указанного способа в качестве метода фундаментального класса `overlong [4]`, дана оценка требуемых вычислительных ресурсов при его использовании.

3. Операция деления

Чтобы разделить число $\boldsymbol{u}$ на число $\boldsymbol{v}$, можно сначала найти достаточно точное приближение к числу $1/\boldsymbol{v}$, затем умножить его на $\boldsymbol{u}$, что даст приближение к $\boldsymbol{u}/\boldsymbol{v}$. Если L_u и L_v – длины операндов, то длина целочисленного ответа будет не более $L_u – L_v + 1$. Число $1/\boldsymbol{v}$ содержит не менее $L_v + 1$ значащих нулей в старших разрядах, кроме того для получения правильного результата деления оно должно содержать еще не менее $L_u – L_v + 1$ значащих цифр. Таким образом, необходимая и достаточная точность вычисления величины $1/\boldsymbol{v}$ составляет величину $b^{-(L_u+2)}$, где b – основание системы счисления.

Применение метода Ньютона к задаче нахождения корня уравнения $\boldsymbol{f(x) = v – 1/x}$ состоит в последовательном вычислении $\boldsymbol{x_{k+1} = (2 – v\, x_k)x_k}$, $k = 0. 1, 2, \ldots$, где $\boldsymbol{x_0}$ – начальное приближение, вычисленное с достаточной точностью. При $x \geq 1$ функция $\boldsymbol{f(x)}$ является дважды непрерывно дифференцируемой и строго выпуклой. В этом случае метод Ньютона обладает квадратичной скоростью сходимости, т.е. количество значащих разрядов после выполнения очередной итерации будет удваиваться.

Легко проверить, что $x_0 = \left\lfloor \left(b^5 - 1\right) / \left(b^2 v_1 + b \cdot v_2 + v_3\right) \right\rfloor \cdot b^{-L_v - 6}$, где v_1, v_2, v_3 – старшие разряды числа $\boldsymbol{v}$, является приближением величины $1/\boldsymbol{v}$ с точностью не хуже $b^{-(L_v+2)}$. Таким образом, потребуется выполнить по методу Ньютона не более $\left\lceil \log_2 \left(\frac{L_u + 3}{L_v + 2} \right) \right\rceil$ итераций.

Изложенное выше позволяет предложить редакцию метода Ньютона для получения обратной величины с использованием только операций целочисленной арифметики. Реализация алгоритма как метода фундаментального класса overlong [4] представлена на рис. 1.

```
#include "overlong.h"

overlong overlong::
        Division(const overlong& b){
if (b.leng<3)
          return operator/(*this,b);

R1: \\Начальное приближение
overlong z;
z.leng = 6;
z.d = new unsigned short[z.leng];
z.d[0] = 0;  z.d[1] = 0;
z.d[2] = 0;  z.d[3] = 0;
z.d[4] = 0;  z.d[5] = 1;
overlong temp;
temp.leng = 3;
temp.d = new unsigned
                   short[temp.leng];
for (int i=0;i<3;i++)
 temp.d[2-i] = b.d[b.leng-i-1];

z/=temp; int newL = z.nsd();

if (newL<z.leng){
   unsigned short* newD =
            new unsigned short[newL];
   for (int i=0;i<newL;i++)
                   newD[i] = z.d[i];
   delete[] z.d;  z.d = newD;
   z.leng = newL;
   }

R2: \\Итерация по Ньютону
int k = 1, n = b.leng;
while (k<n) {
  overlong z_sq = z*z;
  overlong vk;
  vk.leng = 2*k+3;
  vk.d = new unsigned
                      short[vk.leng];
  if (vk.leng<=n)
   for (int i=0;i<vk.leng;i++)
      vk.d[vk.leng-i-1]=b.d[n-i-1];
else {
    for (int i=0;i<n;i++)
       vk.d[vk.leng-i-1]=b.d[n-i-1];
    for (int i=n;i<vk.leng;i++)
                vk.d[vk.leng-i-1]=0;
   }

  vk*=z_sq;
  overlong newZ;
  newZ.leng = z.leng+k;
  newZ.d = new unsigned
                     short[newZ.leng];

  for (int i=0;i<z.leng;i++)
              newZ.d[newZ.leng-i-1]=
                    z.d[z.leng-i-1];

  for (int i=z.leng;i<newZ.leng;i++)
           newZ.d[newZ.leng-i-1] = 0;

  newZ*=(short unsigned)2;
  unsigned short* vk_old_ptr = vk.d;
  int vk_old_leng = vk.leng;
  vk.d += vk_old_leng - newZ.leng;
  vk.leng = newZ.leng;
  newZ-=vk;
  vk.d = vk_old_ptr;
  vk.leng = vk_old_leng;
  z = newZ;
  k*=2;
 }

R3: \\Завершение
 overlong res;
 temp = (*this)*z;
 int resL = temp.leng-k-1-n;
 res.leng = resL;
 res.d =
        new unsigned short[res.leng];
 for (int i=0;i<res.leng;i++)
  res.d[res.leng-i-1]=
              temp.d[temp.leng-i-1];

     return res;
}
```

Рис. 1. Масштабируемый метод целочисленного деления

Оценим необходимые вычислительные ресурсы. Каждая операция алгоритма с длинными числами является либо умножением либо вычитанием ($L_u - L_v + 1$)-разрядных чисел. Как следует из табл. 1 для этого потребуется не более $2(L_u - L_v + 1)$ параллельных процессов. Каждая итерация требует одно вычитание и два умножения чисел с разрядностью не более $(L_u - L_v + 1)$.

Следовательно в соответствии с табл. 1, время выполнения одной итерации не будет превосходить $2(t_M + 3t_A + 2t_A \log_2 L)$, а время выполнения всей операции –

$$2\left\lceil \log_2\left(\frac{L_u+3}{L_v+2}\right)\right\rceil (t_M + 3t_A + 2t_A \log_2 L).$$

4. Заключение

Таким образом, для выполнения всех низкоуровневых операций длинной арифметики возможно эффективное использование многопроцессорных гетерогенных систем. Это позволит освободить центральный процессор для программирования высокоуровневых проблем, требующих доказательных вычислений.

Литература

1. Кнут, Д. Искусство программирования для ЭВМ. т.2. Получисленные алгоритмы / Д. Кнут, пер. с англ. – М:Наука. – 1985. – С. 250-268.
2. Качко Е.Г./Распараллеливание алгоритмов умножения чисел многократной точности / Е.Г. Качко // Параллельные вычислительные технологии (ПаВТ'2011): труды международной научной конференции – Челябинск: Издательский центр ЮУрГУ, 2011. – 730 с. – URL:http://omega.sp.susu.ac.ru/books/conference/PaVT2011 c.509-515.
3. Панюков А.В., Лесовой С.Ю. Применение массивно-параллельных вычислений для реализации основных операций целочисленной арифметики / А.В. Панюков, С.Ю. Лесовой // Высокопроизводительные параллельные вычисления на кластерных системах: материалы X международной конференции: Изд-во ПГТУ, 2010. – С. 77-84.
4. А.В. Панюков, М.И. Германенко, В.В. Горбик. Библиотека классов "Exact Computational" / Свидетельство о государственной регистрации программы для ЭВМ № 2009612777 от 29 мая 2009г. // Программы для ЭВМ, базы данных, топологии интегральных микросхем. Официальный бюллетень Российского агентства по патентам и товарным знакам. – № 3. – 2009. – С. 251.