Электронные и ионные пучки

УДК 004.421.2

Практика распараллеливания вычислений при моделировании электронно-оптических систем

В.С. Иванов

В этой статье рассмотрены теоретические основы параллельных вычислений и приведены способы распараллеливания линейных алгоритмов. На примере алгоритма расчёта потенциала в рабочей области электронно-оптической системы рассмотрена практическая реализация распараллеливания, приведен сравнительный анализ результатов тестирования на различных ЭВМ.

PACS: 41.85.Lc
Ключевые слова: электронно-оптическая система, ЭОС, параллельные вычисления, распараллеливание, многопоточность.

Введение

В процессе моделирования электронно-оптических систем (ЭОС) большое время занимают расчёты электрического поля и траекторий движения частиц, которое для малогабаритных систем, требующих большой точности, может достигать нескольких суток. Поэтому большое значение имеет оптимизация алгоритмов расчёта для максимального использования мощности современных многопроцессорных ЭВМ. Значительный прирост производительности в данном случае даёт распараллеливание линейных вычислений.

Математические основы параллельных вычислений

Параллельные вычисления возможны тогда, когда отсутствует необходимость в завершении предыдущей операции для начала следующей. В качестве примера можно взять следующее выражение: $5 \cdot 7 + 3 \cdot 13$. Для того, чтобы произвести второе умножение, не требуется знать результата первого, следовательно, оба умножения можно произвести параллельно, и только после этого произвести сложение. Очевидно, что не каждое вычисление можно распараллелить. Выражение $(5 \cdot 7 \cdot 3) + 13$ можно вычислить только последовательно, а именно, сначала первое умножение, затем второе, и только после этого – сложение.

В основе анализа распараллеливаемости алгоритмов лежит исследование зависимостей по данным между операциями. Если от результата одной операции зависит результат другой, то вторая называется зависимой по данным от первой. Совокупность всех зависимостей по данным представляет собой ориентированный граф без циклов, в котором вершины – операции, а рёбра – зависимости. Длина самого протяжённого пути по этому графу называется минимальной высотой алгоритма и определяет минимально возможное время, за которое теоретически возможно выполнить вычисления по алгоритму. Практически же, это время не всегда достижимо по той причине, что может потребоваться параллельное вычисление очень большого количества операций, что может оказаться неосуществимым на выделенных вычислительных ресурсах или потребуются очень большие накладки на синхронизацию параллельных вычислительных потоков.

При реализации алгоритма на реальной системе операции распределяются в группы, или кортежи операций, выполняющихся параллельно в один момент времени. Высота алгоритма – количество таких кортежей – определяет, сколько времени потребуется на выполнение.

Реализация параллельных вычислений

Существует два основных подхода к распараллеливанию вычислений в микропроцессорных системах, называемые, соответственно, однопоточным и многопоточным параллелизмом. Различие заключается в использовании одного или нескольких потоков исполнения для параллельных вычислений.

- **Однопоточный параллелизм** заключается в параллельном выполнении операций внутри одного потока исполнения. Возможность однопоточного параллелизма определяется архитектурой микропроцессора, а конкретно, его способностью считывать из памяти и исполнять одновременно несколько операций.
- **Многопоточный параллелизм** – использование нескольких потоков для достижения параллельного исполнения операций. Для того чтобы обеспечить многопоточный параллелизм, необходимо создать систему с несколькими процессорами или процессорными ядрами.

Иванов Владимир Сергеевич, аспирант.
Рязанский Государственный Радиотехнический Университет.
Россия, 390005, Рязань, ул. Гагарина, 59/1.
Тел.: +7 910 616-37-67.
E-mail: vladimir.s.ivanoff@gmail.com

Статья поступила в редакцию 20 декабря 2013 г.

Принципы организации однопоточного и многопоточного параллелизма, их особенности, достоинства и недостатки очень различны и имеют мало общего как в реализации вычислительной системы, так и в построении программного обеспечения.

Однопоточный параллелизм

Как было отмечено выше, однопоточный параллелизм реализуется внутри одного потока исполнения. Возможность однопоточного параллельного исполнения почти целиком лежит на архитектуре микропроцессора – именно она определяет способность микропроцессора выполнять инструкции параллельно.

Однопоточный параллелизм обладает своими достоинствами и недостатками. Достоинства можно сформулировать в следующем виде.

- Отсутствие необходимости синхронизации – все операции выполняются внутри одного потока, и, следовательно, в строго определённой последовательности.
- Отсутствие необходимости поддержки параллелизма на уровне операционной системы.
- Отсутствие необходимости в средствах управления разделяемыми ресурсами (арбитража).

Соответственно, недостатки определяются следующим образом.

- Затруднённость использования в алгоритмах с условными переходами.
- Необходимость адаптации программы для эффективного использования ресурсов микропроцессора, например, при переходе с одной модели на другую.

Существуют следующие методы достижения параллельных вычислений.

- Упаковка данных для групповой обработки единичными инструкциями применением специальных методик. Например, возможно сложить две пары 8-битных данных 16-битной операцией, исключив возможность переполнения. Метод имеет очень ограниченное применение.
- Суперскалярная архитектура. Устройство управления микропроцессора самостоятельно анализирует поток инструкций на предмет возможности параллельного исполнения и управляет несколькими функциональными устройствами.
- Векторная обработка. Микропроцессор имеет инструкции, производящие групповые однотипные операции. Однотипные операнды упаковываются в один векторный регистр. Этот метод аналогичен первому, но обеспечение параллельности лежит на микропроцессорной архитектуре. Векторные регистры, как правило, имеют большую разрядность. Требуется адаптировать программу для использования векторных инструкций или применять оптимизирующий компилятор.
- Микропроцессор с явным параллелизмом. Метод аналогичен второму, но программа для такого процессора содержит явные указания на то, какие операции надо выполнять параллельно. Распараллеливание вычислений в данном случае полностью лежит на программисте или оптимизирующем компиляторе.

Многопоточный параллелизм

Были разработаны специальные технологии для создания многопроцессорных систем, которые позволяли обрабатывать данные параллельно, а, следовательно, существенно быстрее. Соответственно, создавались и операционные системы, поддерживающие многопроцессорные технологии, например, такие, как Solaris (Sun Microsystems), Unix-подобные ОС, Irix (SGI), AIX (IBM), семейство Linux, семейство ОС – Windows.

Поток (thread) % это последовательность инструкций, выполняемых в пределах контекста процесса. Слово «многопоточные» подразумевает содержание множества управляемых потоков. Традиционный процесс содержит один управляемый поток. Многопоточный, в свою очередь, содержит множество потоков, которые выполняются независимо. Так как каждый поток выполняется независимо, то распараллеливание кода программы приводит к следующим преимуществам.

- Улучшение чувствительности приложения.
- Использование многопроцессорности более эффективно.
- Улучшает структуру программы.
- Использование меньше ресурсов системы.
- Улучшение представления.

В настоящее время наиболее распространёнными являются многопоточные системы. Схема распараллеленного вычисления в данном случае будет выглядеть следующим образом.

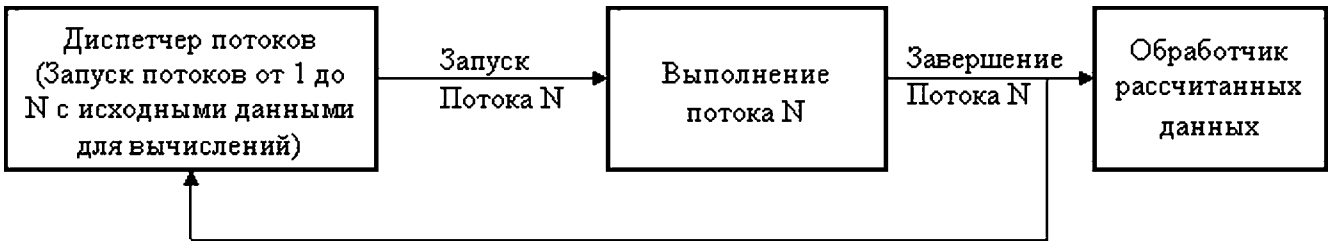

Рис. 1. Схема многопоточного распараллеленного вычисления.

Многопоточный параллелизм при моделировании ЭОС

В работе [1] показана возможность легкого распараллеливания вычисления функции распределения потенциала в рабочем объеме при использовании метода граничных элементов.

Линейный алгоритм расчёта сетки потенциалов $u_{ij}=F(x_j, y_i)$ в прямоугольной области $x\in[Xmin,Xmax]$, $y\in[Ymin,Ymax]$ осуществляется с помощью простейшего алгоритма, выраженного средствами языка PASCAL:

```
for i:=1 to nY do
begin
  Y:=Ymin+Hy*(i-1);
   for j:=1 to nX do
   begin
      X:=Xmin+Hx*(j-1);
       U[i,j]:=F(X,Y);
    end; (*j*)
end; (*i*).
```

Здесь *nX, nY* % количество узлов сетки по направлениям *0X* и *0Y*, соответственно;
Hx=(Xmax-Xmin)/(nX-1),
Hy=(Ymax-Ymin)/(nY-1).

Наиболее трудоёмкой задачей в данном цикле является вычисление функции *F(X,Y)*, представляющей собой численно определяемый интеграл от функции, являющейся комбинацией полных эллиптических интегралов первого и второго родов [6]. Для разделения процесса вычислений выделим как независимую часть цикл по *j*. При запуске такого цикла с различными значениями *Y* получим уменьшение итераций цикла по *i*. Модифицированный алгоритм вычисления сетки потенциалов с помощью двух независимых потоков будет выглядеть следующим образом:

```
Procedure Thread(i,Y)
// Выполнение i-й итерации цикла по j.
Begin
   for j:=1 to nX do
   begin
      X:=Xmin+Hx*(j-1);
      U[i,j]:=F(X,Y);
   end; (*j*)
End; (*Thread*)

(*-modified process-*)
  Imax:= (nY mod 2);
// Количество итераций по i.

  for i:=1 to Imax do
// Основной цикл вычислений.
  begin
     Y:= Ymin + Hy*(2*i-2);
     Thread(i,Y);
// Запуск 1-го потока
     Y:= Ymin + Hy*(2*i-1);
     Thread(i,Y);
// Запуск 2-го потока
     Wait($DEC0DE);
  end; (*i*)
```

В основном цикле асинхронный запуск процедуры Thread(i,Y) происходит дважды. Таким образом, количество итераций уменьшается вдвое. Количество потоков может варьироваться. Наиболее эффективным с точки зрения производительности будет случай, когда их количество будет равно количеству процессоров в системе.

Тестирование

Сравним производительность алгоритма в различных системах. На рис. 2 представлены результаты тестирования на двух 2-ядерных ЭВМ разных архитектур. Соответственно, на рис. 3 отображены результаты тестирования на четырёхпроцессорной ЭВМ.

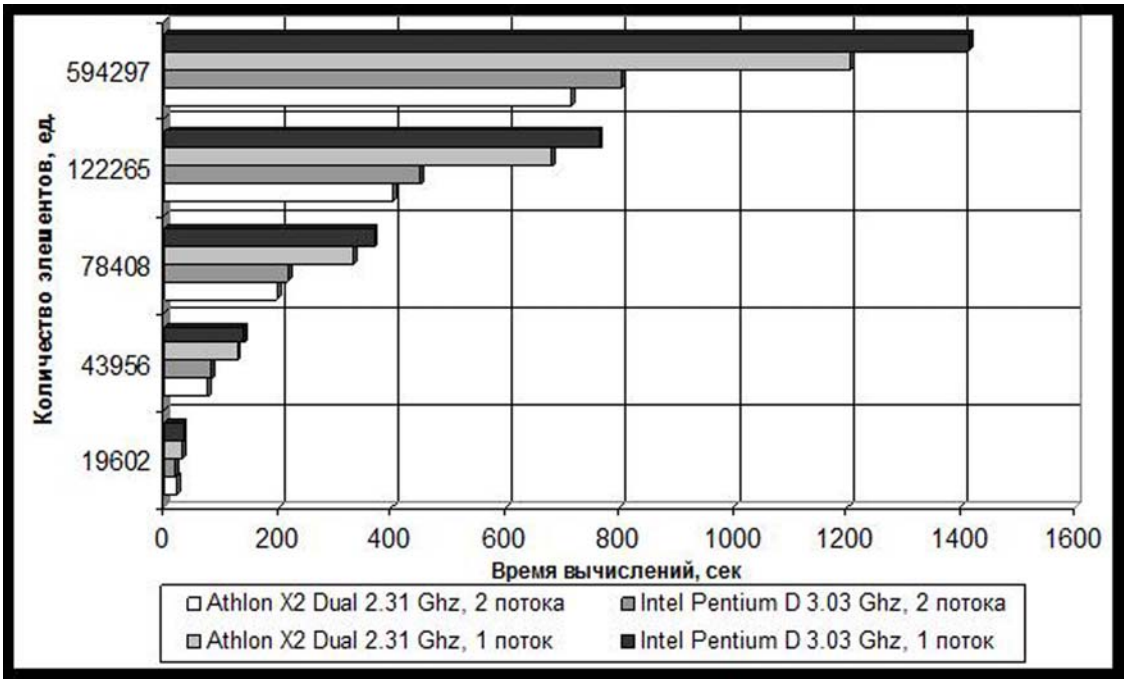

Рис. 2. Результаты тестирования на двух 2-ядерных ЭВМ разных архитектур.

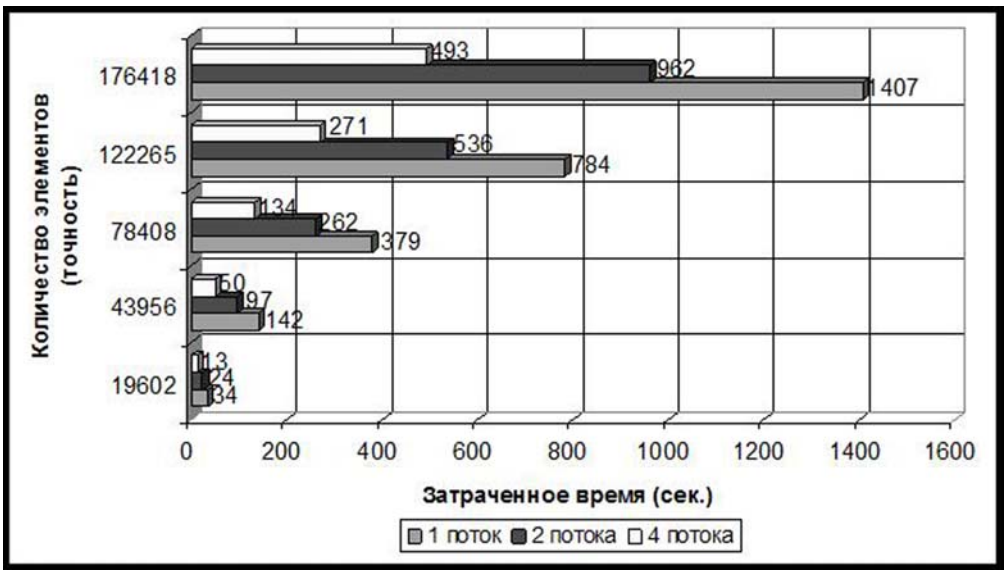

Рис. 3. Результаты тестирования на 4-ядерной ЭВМ.

В последнем случае конфигурация тестового стенда следующая.

- Процессор: Intel Xeon, 2.6 GHz. Количество ядер в процессоре: 4. ОС: Windows 2003 Server, SP2. ОЗУ: 4 Gb.

Заключение

На данный момент существует несколько методов распараллеливания вычислений. В работе была рассмотрена реализация многопоточного параллелизма при решении задач расчёта потенциала методом граничных элементов. Результаты тестирования показали эффективность данного подхода.

Литература

1. *Гуров В.С., Дягилев А.А., Иванов В.С., Трубицын А.А.* // Вестник РГРТУ, 2009, № 27
2. *Власова Е.А., Зарубин В.С, Кувыркин Г.Н.* Приближенные методы математической физики. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2001.
3. *Таненбаум Э.* Современные операционные системы. – СПб: Питер, 2011
4. *Lynch Nancy A.* Distributed Algorithms. – Morgan Kauffman, 2010
5. *Garg Vijay K.,* Elements of Distributed Computing – Wiley-IEEE Press, 2002.
6. *Трубицын А.А.* // Прикладная физика. 2008. № 2. С. 56

Practical aspects of the multithreaded calculations at EOS modeling

V. S. Ivanov

Ryazan State Radio Engineering University
59/1 Gagarin str., Ryazan, 390005, Russia
E-mail: vladimir.s.ivanoff@gmail.com

Received April 5, 2014

EOS modeling time can be reduced by converting algorithms from linear to multithreaded. The theoretical foundations of parallelism, practical realization of multithreaded electric field calculation algorithm and it detailed testing on different CPUs is shown in this paper.

PACS: 41.85.Lc
Keywords: particle beam focusing, CPO, charged particle optics, multithreading, parallelism.

References

1. V. S. Gurov, A. A. Dyagilev, V. S. Ivanov, and A. A. Trubitsin, Bull. Ryazan State Radio Engineer. Univer., No. 27 (2009).
2. E. A. Vlasova, V. S. Zarubin, and G. N. Kuvirkin, Approximate Methods of Mathematical Physics (Bauman MGTU, Moscow, 2001) [in Russian].
3. E. Tanenbaum, *Modern Operating Systems* (Piter, SPb, 2011) [in Russian].
4. Nancy A. Lynch, *Distributed Algorithms* (Morgan Kauffman, 2010).
5. Vijay K. Garg, *Elements of Distributed Computing* (Wiley-IEEE Press, 2002).
6. A. A. Trubitsin, Prikladnaya Fizika, No. 2, 56 (2008).