ЛАБОРАТОРНАЯ РАБОТА №4 СИММЕТРИЧНАЯ МУЛЬТИПРОЦЕССОРНАЯ ОБРАБОТКА

Цель работы – знакомство с особенностями многопоточной обработки информации на многоядерных процессорах под управлением ОС MS Windows и методом оценки трудоемкости алгоритмов

1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1 Мультипроцессорная и симметричная мультипроцессорная обработка (SMP)

Мультипроцессорная обработка - способ организации вычислительного процесса в системах с несколькими процессорами (ядрами процессора), при котором несколько потоков могут одновременно выполняться на разных процессорах (ядрах) системы.

Мультипроцессорные системы часто характеризуют либо как симметричные, либо как несимметричные. При этом нужно определять, к какому аспекту мультипроцессорной системы относится эта характеристика - к типу архитектуры или к способу организации вычислительного процесса.

Симметричная архитектура мультипроцессорной системы предполагает однородность всех процессоров и единообразие включения процессоров в общую схему мультипроцессорной системы. Традиционные симметричные мультипроцессорные конфигурации разделяют общую оперативную память между всеми процессорами (ядрами процессоров).

Масштабируемость (возможность наращивания числа процессоров) в симметричных системах ограничена вследствие того, что все они пользуются одной оперативной памятью и должны располагаться в одном корпусе. Такая конструкция, называемая *масштабируемой по вертикали*.

В симметричных архитектурах обеспечивается достаточно высокая производительность для тех приложений, в которых несколько задач должны активно взаимодействовать между собой.

В асимметричной архитектуре разные процессоры могут отличаться как своими характеристиками, так и функциональной ролью, которая поручается им в системе. Функциональная неоднородность в асимметричных архитектурах влечет за собой структурные отличия во фрагментах системы, содержащих разные процессоры системы. Масштабирование в асимметричной архитектуре реализуется иначе, чем в симметричной. Так как требование единого корпуса отсутствует, система может состоять из нескольких устройств, каждое из которых содержит один или несколько процессоров. Это масштабирование по *горизонтали.* Каждое такое устройство называется *кластером*, а вся

мультипроцессорная система - кластерной.

Асимметричное мультипроцессирование является наиболее простым способом организации вычислительного процесса в системах с несколькими процессорами. Этот способ часто называют также «ведущий-ведомый». Асимметричная организация вычислительного процесса может быть реализована как для симметричной мультипроцессорной архитектуры, так и для несимметричной.

Симметричное мультипроцессирование как способ организации вычислительного процесса может быть реализовано в системах только с симметричной мультипроцессорной архитектурой. Симметричное мультипроцессирование реализуется общей для всех процессоров операционной системой. При симметричной организации все процессоры равноправно участвуют и в управлении вычислительным процессом, и в выполнении прикладных задач.

Наиболее распространенной целью объединения процессоров является симметричная мультипроцессорная обработка (SMP). В системе SMP каждый процессор решает свою задачу, порученную ему операционной системой. В документации Intel симметрия рассматривается в двух аспектах:

- симметрия памяти — все процессоры пользуются общей памятью и одной копией ОС;
- симметрия ввода-вывода — все процессоры разделяют общие устройства ввода-вывода и общие контроллеры прерываний.

Система может быть симметричной по памяти, но асимметричной по прерываниям от ввода-вывода, если для обслуживания этих прерываний выделяется собственный процессор. В x86 симметрию по прерываниям обеспечивают контроллеры APIC. Аппаратная (физическая) реализация SMP может быть различной:

- объединение нескольких физических процессоров на одной локальной шине — процессоры Pentium, P6, Pentium 4;
- размещение на одном кристалле нескольких логических процессоров с разделяемыми операционными блоками — «гиперпотоковые» (hyperthreading) модели Pentium 4;
- размещение на одном кристалле нескольких независимых процессорных ядер с разделяемым вторичным кэшем — мультиядерные модели Pentium 4.

Применение SMP требует поддержки со стороны BIOS, ОС и приложений (чтобы работать быстрее, они должны быть многопоточными). Поддержку SMP имеют такие ОС, как Microsoft Windows и различные диалекты Unix и Linux.

Несколько лет назад цена мультипроцессорных версий ОС, как правило, была значительно выше цены соответствующих однопроцессорных версий, что становилось препятствием к применению

гиперпотоковых и мультиядерных процессоров в системных платах рабочих станций. Теперь число процессоров, на которое лицензируется ОС, соответствует числу физических процессоров. Это открывает возможности широкого распараллеливания на уровне процессоров.

1.2 Оценка достигаемого выигрыша в производительности – закон Амдала

Для количественной оценки выигрыша в производительности ПК при параллельной работе нескольких ядер обычно используется закон Дж. Амдала (1967 г).

Закон Амдала описывает максимальный теоретический выигрыш в производительности параллельного решения по отношению к лучшему последовательному решению [1]

$$V = \frac{1}{S + (1 - S)/n}$$

В данном уравнении V – выигрыш в производительности при использовании n ядер центрального процессора, S – время, потраченное на выполнение последовательной части параллельной версии.

При n=1 (одно ядро) ускорения нет. Если используется два ядра, которые половину всей работы выполняют параллельно, S=0,5 и V= 2 / 1,5 = 1,33. В случае выполнения всей работы двумя ядрами параллельно максимально возможный теоретический выигрыш равен 2.

1.3 Оценка трудоемкости алгоритма

Цель анализа трудоёмкости алгоритма - нахождение оптимального алгоритма для решения задачи. В качестве критерия оптимальности алгоритма выбирается трудоемкость алгоритма, определяемая как количество операций, которые необходимо выполнить для решения задачи с помощью данного алгоритма. Функцией трудоемкости называется соотношение, связывающее размер данные алгоритма с количеством элементарных операций, необходимых для получения решения задачи с помощью данного алгоритма.

Трудоёмкость алгоритмов по-разному зависит от входных данных. Для некоторых алгоритмов трудоемкость зависит только от объёма данных, для других алгоритмов — от значений данных. Трудоёмкость многих алгоритмов может в той или иной мере зависеть от всех перечисленных выше факторов. Одним из упрощенных методов анализа, используемых на практике, является *асимптотический анализ трудоемкости алгоритмов*. Целью анализа является сравнение затрат времени различными алгоритмами, предназначенными для решения одной и той же задачи, при больших объёмах входных данных. Используемая в

асимптотическом анализе оценка функции трудоёмкости, называемая **сложностью алгоритма**, позволяет определить, как быстро растет трудоёмкость алгоритма с увеличением объёма данных.

Например, трудоемкость алгоритма сложения векторов A(n) и B(n) равна O(n), потому что количество операций сложения равно количеству элементов векторов. Трудоемкость алгоритма умножения квадратных матриц равна $O(n^3)$. Реализующая алгоритм программа умножения матриц содержит 3 вложенных арифметических цикла.

2 МЕТОДИКА ВЫПОЛНЕНИЯ

1. В работе оценивается трудоемкость простейших алгоритмов и эффективность их параллельного выполнения на многоядерных процессорах под управлением ОС MS Windows 7, поддерживающей SMP. Для оценки трудоемкости применяется оценка времени выполнения реализующей алгоритм программы на одном или нескольких ядрах ЦП.
2. Для управления количеством используемых процессорных ядер используется диспетчер задач (контекстное меню, пункт *задать соответствие*). Каждая программа должна быть многократно запущена на одном, двух, трех, четырех и более (сколько есть у ЦП) ядрах.
3. Так как время выполнения программы в многозадачной ОС MS Windows зависит от нескольких факторов, для оценки времени следует выполнить программу 5-7 раз с неизменными начальными условиями и в качестве оценки времени выполнения выбрать наименьшее значение.
4. Размер обрабатываемого массива следует задавать в пределах 100-500, при этом время выполнения приложения не должно быть менее 500 мсек. Количество различных значений должно лежать в пределах от 4 до 10.
5. Наименование программы, количество используемых ядер ЦП, количество потоков программы (по данным диспетчера задач), размер обрабатываемого массива и время выполнения при каждом запуске записать в таблицу (форма таблицы произвольная).
6. Полученные результаты обработать: вычислить реальное значение выигрыша по производительности и сравнить со значением выигрыша, найденного по закону Амдала. Сравнить характер изменения оценок реального времени выполнения программы (при различных размерах обрабатываемого массива) и асимптотической оценки трудоемкости алгоритма, который реализует исследуемая программа. Для нахождения оценок параметров модели, описывающей зависимость времени выполнения программы от

размера обрабатываемых данных, рекомендуется использовать метод наименьших квадратов.

7. Каждая бригада должна выполнить исследование алгоритма УМНОЖЕНИЯ МАТРИЦ и алгоритма, указанного в индивидуальном задании – таблица 1. Исследование алгоритма состоит в последовательном выполнении пунктов 1-6.
   Текст программы умножения матриц (на языке PascalABC.Net) приведен ниже. Распараллеливание циклов For реализуется с помощью директив OpenMP.

```
uses Arrays;
procedure ParallelMult(a,b,c: array [,] of real; n: integer);
begin
  {$omp parallel for }
  for var i:=0 to n-1 do
    for var j:=0 to n-1 do
    begin
      c[i,j]:=0;
      for var l:=0 to n-1 do
        c[i,j]:=c[i,j]+a[i,l]*b[l,j];
    end;
end;

procedure Mult(a,b,c: array [,] of real; n: integer);
begin
  for var i:=0 to n-1 do
    for var j:=0 to n-1 do
    begin
      c[i,j]:=0;
      for var l:=0 to n-1 do
        c[i,j]:=c[i,j]+a[i,l]*b[l,j];
    end;
end;

const n = 300;

begin
  var a := Arrays.CreateRandomRealMatrix(n,n);
  var b := Arrays.CreateRandomRealMatrix(n,n);
  var c := new real[n,n];
  ParallelMult(a,b,c,n);
  writeln('Параллельное перемножение матриц: ',Milliseconds,' миллисекунд');
```

```
  var d := Milliseconds;
  Mult(a,b,c,n);
  writeln('Последовательное перемножение матриц: ',Milliseconds-d,' миллисекунд');
end.
```

При отсутствии на ПК платформы PascalABC.Net для выполнения работы используется файл multMatrix.exe из каталога лабораторной работы.

Таблица 1.

Индивидуальные задания для бригад

№№ бригад	Алгоритм
1	Быстрая сортировка QuickSort
2	Ханойские башни Hanoi
3	Нахождение суммы простых чисел SumOfPrime
4	Сложение матриц
5	Сортировка методом пузырька

3 ОТЧЕТ О РАБОТЕ

Готовится в письменном виде один на бригаду. Содержание отчета:

1. Таблицы, содержащие результаты выполнения п. 2 (времена выполнения программ на одном, двух и более ядрах ЦП)
2. Результаты обработки данных построенных в п.1 таблиц – теоретическая и эмпирическая оценки выигрыша в производительности.
3. Оценки трудоемкости исследованных алгоритмов.
4. Вывод

4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Симметричная и асимметричная архитектуры аппаратных и программных средств.
2. Достоинства симметричной архитектуры.
3. Понятие SMP
4. Закон Амдала
5. Трудоемкость алгоритма
6. Трудоемкость алгоритмов умножения матриц, сложения матриц и сортировки массива методом пузырька.
7. Трудоемкость алгоритма быстрой сортировки