Московский государственный университет им. М.В. Ломоносова

Введение в методы параллельных вычислений

Разработчик:
А.В. Старченко, д.ф.-м.н., профессор
E-mail: starch@math.tsu.ru
Томский государственный университет

Направление 010300.68
«Фундаментальная информатика и информационные технологии»

Проект комиссии Президента по модернизации и техническому развитию экономики России «Создание системы подготовки высококвалифицированных кадров в области суперкомпьютерных технологий и специализированного программного обеспечения»

Содержание курса

- Введение
- Рекуррентные формулы
- Параллельные вычисления определенных и кратных интегралов
- Умножение матрицы на вектор. Умножение матриц
- Прямые методы решения систем линейных уравнений на многопроцессорных системах Организация межпроцессорных обменов
- Трехдиагональные системы. Параллельная реализация прямых методов решения систем линейных уравнений
- Параллельная реализация итерационных методов решения СЛАУ
- Параллельная реализация быстрого преобразования Фурье

Содержание лекции

- Производительность компьютерной программы
- Степень параллелизма компьютера и степень параллелизма алгоритма
- Пример (суммирование двух векторов и вычисление скалярного произведения)
- Показатели оценки эффективности параллельного алгоритма и программы
- Пример (оценка ускорения параллельных алгоритмов суммирование двух векторов и вычисление скалярного произведения)
- Формальная модель ускорения. Закон Амдала
- Основные этапы разработки параллельных алгоритмов

Производительность компьютерной программы

- Производительность программы - это величина, обратная процессорному времени, затрачиваемому на выполнение программы.
- Для параллельных компьютеров минимизация времени выполнения программы не есть синоним минимального числа арифметических операций.
- Максимальная производительность достигается при соответствии программы архитектуре компьютера.
- Автоматизированные средства распараллеливания последовательных программ – это хорошо, но роль математика-программиста очень важна в создании высокопроизводительного программного обеспечения.

Степень параллелизма компьютера и степень параллелизма алгоритма

- Степень параллелизма компьютера – количество информации, которое может быть обработано одновременно.
- Степень параллелизма вычислительного алгоритма – число арифметических операций, которое можно выполнить независимо, т.е. параллельно.
- Главная задача программиста-аналитика – найти метод решения задачи, который отражает наилучшее соответствие между параллелизмом алгоритма и параллелизмом компьютера.

Пример (расчет суммы двух векторов и их скалярного произведения)

- **Задача 1**. Дано $\vec{a}, \vec{b} \in \mathbb{R}^n$, найти $\vec{c} = \vec{a} + \vec{b}$.
  Алгоритм: $c_i = a_i + b_i, i = 1,...,n$.
  Степень параллелизма алгоритма -n.

- **Задача 2**. Дано $\vec{a}, \vec{b} \in \mathbb{R}^n$, найти $d = (\vec{a}, \vec{b})$.
  Алгоритм: $(1)\ e_i = a_i \times b_i, i = 1,...,n;\ (2)\ d = \sum_{i=1}^{n} e_i$
  Степень параллелизма алгоритма меняется:
  на этапе (1) = n
  на этапе (2) при обычной схеме суммирования = 1.
  Требуется модификация определения степени параллелизма алгоритма!

Пример (расчет суммы двух векторов и скалярного произведения)

- Определение: Средней степенью параллелизма численного алгоритма называется отношение общего числа арифметических операций алгоритма к количеству его этапов.
- В Задаче 1 она равна n, а в Задаче 2 - $(2\times n)/(1+n)$.
- При использовании алгоритма сдваивания при выполнении этапа (2) Задачи 2 одновременно будет выполняться $\log_2 n$ операций сложения и средняя степень параллелизма будет $(2\times n)/(1+\log_2 n)$.

Алгоритм сдваивания: $((e_1+e_2)+(e_3+e_4))+((e_5+e_6)+(e_7+e_8))$

Обычная схема: $(((((((e_1)+e_2)+e_3)+e_4)+e_5)+e_6)+e_7)+e_8)$

Ускорение и эффективность

- Ускорение параллельной программы на ***p***-процессорной системе $S_p = T_1 / T_p$, где
  T_1 – время выполнения программы на одном процессоре,
  T_p – время выполнения программы на системе из ***p*** процессоров.
- Эффективность параллельной программы на ***p***-процессорной системе $E_p = S_p / p$.
- Эти показатели могут также использоваться для приближенной предварительной оценки ускорения и эффективности разрабатываемого параллельного алгоритма.

Пример (оценка ускорения параллельных алгоритмов, *n/p*>>1)

- Для Задачи 1

  $T_1 \approx t_{add}\times n,\ \ T_p \approx t_{add}\times n/p \Rightarrow S_p = T_1/T_p \approx p$
- Для Задачи 2

  $T_1 \approx (t_{add}+t_{mult})\times n,\ \ \alpha = (t_{add}+t_{comm})/(t_{add}+t_{mult}) \gg 1$

  с использованием последовательной схемы

  $T_p \approx (t_{add}+t_{mult})\times n/p + (t_{add}+t_{comm})\times p \Rightarrow$

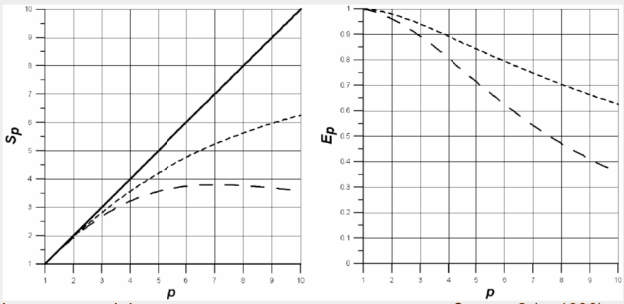
  $S_p \approx p/(1+\alpha\times p^2/n)$

  с использованием алгоритма сдваивания

  $T_p \approx (t_{add}+t_{mult})\times n/p + (t_{add}+t_{comm})\times \log_2 p \Rightarrow$

  $S_p \approx p/(1+\alpha\times p\times \log_2 p/n)$

Пример (оценка ускорения параллельных алгоритмов)



Ускорение и эффективность параллельного алгоритма Задачи 2 (*n*=1000) при использовании последовательного алгоритма суммирования (крупный штрих) и алгоритма сдваивания (мелкий штрих). α = 20

Формальная модель ускорения

$$S_p = \frac{T_1}{(\alpha_1 + \alpha_2/k + \alpha_3/p)\times T_1 + T_{comm}}$$

- $\alpha_1 \ge 0$ - доля операций в параллельной программе, выполняемых последовательно, $(\alpha_1+\alpha_2+\alpha_3=1)$
- $\alpha_2 \ge 0$ - доля операций в параллельной программе, выполняемых со средней степенью параллелизма ***k<p***,
- $\alpha_3 \ge 0$ - доля операций в параллельной программе, выполняемых с максимальной степенью параллелизма ***p***,
- T_1 - время исполнения одним процессором наиболее эффективной последовательной программной версии алгоритма,
- T_{comm}- время, необходимое для подготовки данных (передачи их на задействованные процессоры) для проведения параллельных вычислений.

Формальная модель ускорения

- Случай 1: $\alpha_1 = \alpha_2 = 0, \alpha_3 = 1, T_{comm} = 0 \Rightarrow S_p = p$
- Случай 2: $\alpha_1 = \alpha_3 = 0, \alpha_2 = 1, T_{comm} = 0 \Rightarrow S_p = k < p$
- Случай 3: $\alpha_2 = 0, T_{comm} = 0, \alpha_3 = 1-\alpha_1, \Rightarrow$

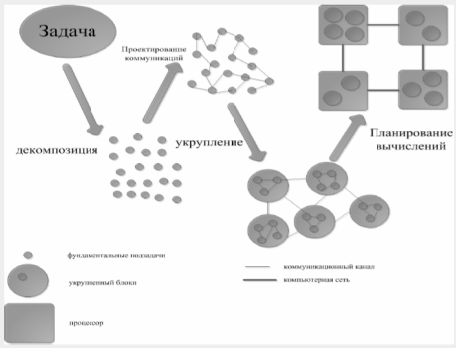
  $S_p = 1/(\alpha_1 + (1-\alpha_1)/p) < 1/\alpha_1$

  Закон Амдала (Amdahl's Law): независимо от количества используемых процессоров и даже при игнорировании затрат на подготовку данных для организации параллельных вычислений ускорение ограничено величиной, обратной доле последовательных операций программы (алгоритма)
- Случай 4: $T_{comm} > T_1, \forall \alpha_i \ \sim \ S_p < 1$

Основные этапы разработки параллельных программ

- Декомпозиция – дробление задачи на более мелкие фрагменты (структуры данных и фундаментальные мелкозернистые подзадачи), которые могут выполняться одновременно.
- Проектирование коммуникаций – установление связей между фундаментальными подзадачами.
- Укрупнение – объединение мелкозернистых подзадач в крупные блоки для уменьшения коммуникационных затрат.
- Планирование вычислений – назначение укрупненных блоков задачи процессорам с учетом снижения коммуникационных затрат и эффективности параллельных вычислений.

Основные этапы разработки параллельных программ



Требования к декомпозиции

- Количество фундаментальных подзадач должно быть, по крайней мере, на порядок больше планируемого числа используемых процессоров, а сами подзадачи в перспективе должны иметь примерно одинаковый размер.
- Пересечение подзадач должно быть сведено к минимуму, чтобы избежать дублирования вычислений.
- Декомпозиция должна быть такой, чтобы при увеличении размера задачи увеличивалось бы количество фундаментальных подзадач, а не размер каждой подзадачи.

Требования к декомпозиции

- Размер фундаментальной подзадачи определяется степенью детализации распараллеливаемого алгоритма, которая характеризуется количеством операций в подзадаче.
- Если в алгоритме можно выделить подзадачи, в которых выполняется лишь несколько операций, то говорят о мелкозернистом параллелизме.
- Если подзадачи определяются на уровне процедур, в которых количество операций до 1000, то имеет место среднеблочный параллелизм.
- Крупнозернистый параллелизм алгоритма характеризуется возможностью выделения нескольких крупных задач на уровне отдельных программ, которые могут выполняться независимо на компьютере с параллельной архитектурой, что требует, как правило, поддержки вычислений операционной системой.

Стратегии декомпозиции

- Декомпозиция по данным - фрагментируются данные, с которыми связываются операции алгоритма обработки.
- Функциональная декомпозиция - сначала декомпозируется алгоритм обработки данных, затем сами данные.
- Независимые задачи – весь объем вычислений делится на подзадачи, которые не зависят друг от друга.
- На этом этапе разработки программы для параллельных вычислений не учитываются особенности архитектуры многопроцессорной вычислительной системы.

Проектирование коммуникаций

- На этом этапе устанавливаются коммуникации (связи) между фундаментальными подзадачами.
- Определяется коммуникационная модель передачи данных между подзадачами.
- Выбираются алгоритмы и методы коммуникаций.
- На этом этапе также не рассматривается архитектура суперкомпьютера.

Типы коммуникаций

- Глобальные и локальные
- Структурированные и неструктурированные
- Статические и динамические
- Синхронные и асинхронные

Требования к проектированию коммуникаций

- У каждой подзадачи количество коммуникаций и объем передаваемых по коммуникациям данных были одинаковыми.
- Где это возможно, лучше использовать локальные коммуникации, выполняющиеся одновременно.
- По возможности коммуникации лучше осуществлять одновременно с вычислениями.
- Коммуникации не должны сдерживать одновременное выполнение фундаментальных подзадач.

Укрупнение

- Производится объединение подзадач в более крупные блоки для повышения эффективности создаваемого алгоритма – чтобы, например, снизить коммуникационные затраты или трудоемкость разработки алгоритма.
- Учитывается архитектура многопроцессорной системы, для которой разрабатывается параллельная программа.

Требования к укрупнению

- Количество укрупненных блоков фундаментальных подзадач должно соответствовать числу используемых процессоров (ядер).
- Первыми кандидатами для комбинирования в укрупненные блоки являются фундаментальные подзадачи, которые не могут выполняться одновременно и независимо.
- При укрупнении нужно помнить, что иногда удается избежать коммуникаций за счет дублирования вычислений различными подзадачами или блоками подзадач, тем не менее, должны быть сохранены масштабируемость и производительность программы.

Планирование вычислений

- На этом этапе осуществляется назначение укрупненных подзадач определенным процессорам в соответствии с требованиями снижения коммуникационных затрат и обеспечения параллелизма.
- Особое значение этот этап имеет при проведении вычислений на гетерогенных многопроцессорных системах.
- Стратегия размещения укрупненных блоков подзадач на процессорах строится с учетом основного критерия – минимизации времени выполнения параллельной программы.
- Чаще всего используются стратегии «хозяин/работник», иерархические и децентрализованные схемы.

Заключение

- Определены понятия параллелизма компьютера и параллелизма алгоритма. На примере показано, что важным в разработке эффективных высокопроизводительных программ является соответствие между параллелизмом алгоритма и параллелизмом компьютера.
- Представлены основные показатели эффективности параллельной программы (алгоритма). Сформулированы формальная модель ускорения и закон Амдала.
- Подробно рассмотрены основные этапы разработки параллельной программы.

Вопросы для обсуждения

- В каких единицах измеряется производительность компьютера?
- Что такое степень параллелизма компьютера? Чему она равна для конкретных компьютеров?
- Что такое степень параллелизма алгоритма?
- Как определяется ускорение параллельной программы? В каких пределах оно может меняться?
- Можно ли распространить определение ускорения для параллельного алгоритма? Дайте комментарий.
- На каких математических принципах базируется алгоритм сдваивания? В чем его преимущество перед обычным алгоритмом суммирования?
- Сформулируйте закон Амдала. Каким будет максимальное ускорение параллельной программы, в которой 1% арифметических операций выполняется только одним процессором?
- Пользуясь формальной моделью ускорения, определите условия, при которых будет иметь место максимальный параллелизм, параллелизм среднего уровня и отсутствие ускорения параллельной программы.
- Перечислите основные этапы разработки параллельной программы. Какие из них зависят от архитектуры ЭВМ, какие не зависят?
- Каковы требования к проведению первого этапа построения параллельного алгоритма?
- Чем отличается мелкозернистый параллелизм алгоритма от крупнозернистого?
- В чем отличие функциональной декомпозиции от декомпозиции по данным?
- Перечислите типы коммуникаций. Дайте им краткую характеристику.

Темы заданий для самостоятельной работы

- Оцените ускорение и эффективность параллельного алгоритма получения векторного произведения векторов.
- Оцените ускорение и эффективность параллельного алгоритма умножения матрицы на вектор ***gaxpy*** (y=y+Ax).
- Оцените ускорение и эффективность алгоритма ***saxpy*** (y=y+αx).
- Оцените ускорение и эффективность алгоритма вычисления нормы вектора.

Литература

- *Хокни Р., Джессхоуп К.* Параллельные ЭВМ. Архитектура, программирование и алгоритмы. М: Радио и связь, 1986.
- *Ортега Дж.* Введение в параллельные и векторные методы решения линейных систем. М.: Мир, 1991.
- *Немнюгин С.А., Стесик О.Л.* Параллельное программирование для многопроцессорных вычислительных систем. СПб.: БХВ-Питербург, 2002.
- *Немнюгин С.А.* Средства программирования для многопроцессорных вычислительных систем. – СПб.: СПбГУ, 2007. – 88 с.
- *http://www.cse.illinois.edu/courses/cs554/notes/*
- *Старченко А.В., Есаулов А.О.* Параллельные вычисления на многопроцессорных вычислительных системах. – Томск: Изд-во Том. ун-та, 2002. – 56 с.

Следующая тема

- Введение
- Рекуррентные формулы
- Параллельные вычисления определенных и кратных интегралов
- Умножение матрицы на вектор. Умножение матриц
- Прямые методы решения систем линейных уравнений на многопроцессорных системах Организация межпроцессорных обменов
- Трехдиагональные системы. Параллельная реализация прямых методов решения систем линейных уравнений
- Параллельная реализация итерационных методов решения СЛАУ
- Параллельная реализация быстрого преобразования Фурье