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

Содержание лекции

- Вычисления по рекуррентным формулам
- Задача нахождения частичных сумм последовательности числовых значений
- Последовательный алгоритм и диаграмма маршрутизации
- Каскадная схема суммирования
- Алгоритм сдваивания
- Модифицированная каскадная схема суммирования
- Задача вычисления членов последовательности по линейной рекуррентной формуле первого порядка и ее распараллеливание с помощью метода циклической редукции
- Оценка ускорения параллельного алгоритма

Вычисления по рекуррентным формулам

- **Рекуррентная формула** (от лат. recurrens, родительный падеж recurrentis — возвращающийся), формула приведения, формула, сводящая вычисление *n*-го члена какой-либо последовательности (чаще всего числовой) к вычислению нескольких предыдущих её членов (БСЭ).
- В качестве примеров можно указать итерации, вычисления по рекуррентным формулам в методе прогонки, при решении нестационарных и граничных задач, получение чисел Фибоначчи.
- Вычисления по рекуррентным формулам составляют определённую проблему для компьютера с параллельной архитектурой.

Расчет частичных сумм последовательности чисел

- Рассмотрим задачу нахождения частичных сумм последовательности числовых значений $x_1,\ldots,x_n$

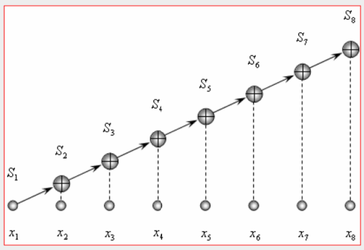
$$S_k=\sum_{i=1}^{k}x_i, 1\le k\le n,$$

- В последовательном алгоритме частичные суммы можно вычислить весьма просто из рекуррентного соотношения

$$S_k=S_{k-1}+x_k, k=1,2,\ldots,n; S_0=0.$$

Последовательный алгоритм и диаграмма маршрутизации

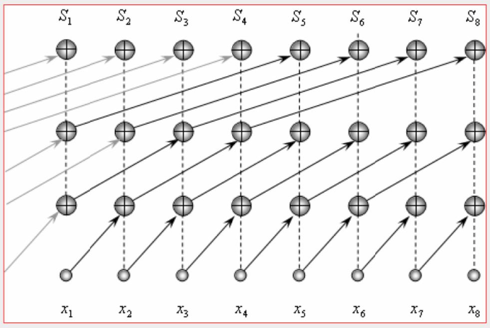
- Соотношение между способом хранения данных, арифметическими операциями и временем можно изобразить на диаграмме маршрутизации.
- Последовательность вычислений перемещается снизу вверх.
- Операции, которые могут выполняться параллельно, показаны на одном и том же вертикальном уровне.



Каскадная схема суммирования

- набор из ***n*** процессорных элементов (ПЭ) сначала загружается данными $\{x_i, i=1,\dots,n\}$, которые должны суммироваться; $S_i = x_i$
- на первом этапе копия S_i из ОЗУ каждого ПЭμ ($\mu=1,\dots,n$) передается соседнему справа ПЭ$\mu+1$ и складывается с S_i этого ПЭ$\mu+1$;
- на следующем этапе процесс повторяется, но уже со сдвигом на две позиции;
- …
- на ***k***-ом этапе сдвиг осуществляется на 2^k позиции;
- …
- после ($\log_2 n$)-го этапа каждый ПЭ содержит требуемые частичные суммы.

Диаграмма маршрутизации каскадной схемы суммирования



Каскадная схема суммирования

- Видно, что метод каскадных сумм требует $\log_2 n$ сложений со степенью параллелизма ***n***, если для обеспечения однородности вычислительной схемы на ПЭ, для которых нет данных слева, отправлять нулевые значения.
- Каждая операция алгоритма каскадных сумм имеет максимально возможную степень параллелизма ***n***.
- Общее число скалярных арифметических операций увеличилось с ***n*** до $n*\log_2 n$, а время выполнения задачи уменьшилось.

Каскадная схема суммирования

- Таким образом, увеличение степени параллелизма на каждом этапе от 1 до ***n*** происходит за счёт существенного увеличения числа скалярных арифметических операций. Например, при ***n*** = 64 – в 6 раз.
- Подсчитаем для рассмотренного алгоритма нахождения частных сумм показатели ускорения и эффективности, не учитывая затраты на пересылку данных.

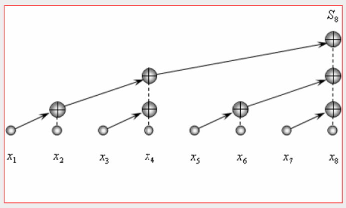
$$S_p = T_1 / T_p \approx n / \log_2 n, (p = n), E_p = S_p / p \approx 1/\log_2 n.$$

Алгоритм сдваивания

- Алгоритм каскадного суммирования частных сумм значительно упрощается в случае, если требуется найти только значение общей суммы S_n:
- на первом этапе копия $S_i = x_i$ из ОЗУ ПЭμ с нечетными индексами копируется направо соседнему ПЭ$\mu+1$ с последующим суммированием;
- далее на ***k***-ом этапе ($k=0,1,2,\dots,(\log_2 n)-1$) копия полученной суммы с ПЭμ ($\mu \bmod 2^{k+1}=2^k$) передается на ПЭ$\mu+2^k$, где она добавляется к имеющейся в памяти сумме.
- После завершения процесса на ПЭ***n*** получается искомый результат.

Диаграмма маршрутизации и псевдокод алгоритма сдваивания

- ***n:=2^q; S:=x;***
- ***For k:=0 to q-1 do***
- ***begin***
- ***If (μ mod 2^{k+1} = 2^k) then***
- ***send(S, μ+2^k);***
- ***If (μ mod 2^{k+1} = 0) then***
- ***begin***
- ***receive(S1, μ-2^k);***
- ***S:=S+S1***
- ***end;***
- ***end;***

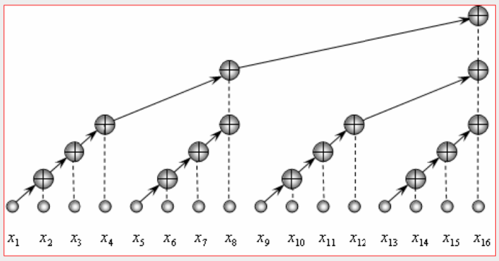


$$S_p \approx n / \log_2 n; (p = n);$$
$$E_p \approx 1/\log_2 n$$

Модифицированная каскадная схема суммирования

- На этапе инициализации все суммируемые значения подразделяются на ***n/log₂n*** групп, в каждой из которых содержится ***log₂n*** элементов.
- Далее в каждой группе вычисляется сумма значений при помощи последовательного алгоритма суммирования.
- На заключительном этапе при суммировании полученных для отдельных групп ***n/log₂n*** значений применяется обычная каскадная схема.

Диаграмма маршрутизации МКСС



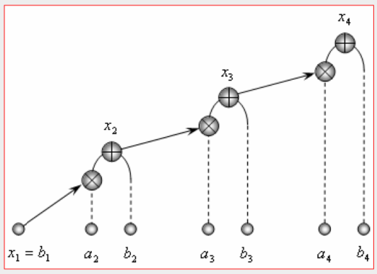
$$T_p \approx t_{add}\left(\log_2 n + \log_2\left(n/\log_2 n\right)\right) < 2t_{add}\log_2 n$$

$$S_p \approx n/(2\log_2 n);\quad E_p \approx 1/2;\quad p = n/\log_2 n$$

Линейная рекуррентная формула

- Элементы числовой последовательности $x_1, x_2, \dots, x_n$ определяются как
  $$x_j = a_j x_{j-1} + b_j,\ j = 1,\dots,n$$
- где $\{a_j\},\{b_j\}$ - известные коэффициенты ($a_1 = 0$).
- При вычислении элементов последовательности требуется $2 \cdot n$ арифметических операций.
- В численном анализе, когда необходимо распараллелить вычислительный алгоритм, широко применяется способ, эквивалентный методу, лежащему в основе каскадной схемы суммирования, и известный как *циклическая редукция*.

Схема вычисления элементов последовательности



Циклическая редукция

- Основная идея этого метода заключается в последовательном построении рекуррентных соотношений между каждым вторым, затем каждым четвертым, затем каждым восьмым и т.д. элементами последовательности за счет исключения промежуточных элементов последовательности.
- Продолжая этот процесс, можно на заключительном этапе (после ***log₂n*** редукций) получить соотношения, в которых ***xⱼ*** будет определяться по явной формуле.

Формулы метода циклической редукции

$$\left.\begin{aligned} x_j &= a_j x_{j-1} + b_j \\ x_{j-1} &= a_{j-1} x_{j-2} + b_{j-1} \end{aligned}\right\}$$

$$x_j = a_j a_{j-1} x_{j-2} + b_j + a_j b_{j-1} = a_j^{(1)} x_{j-2} + b_j^{(1)},$$

где

$$a_j^{(1)} = a_j a_{j-1}$$

$$b_j^{(1)} = b_j + a_j b_{j-1}$$

Формулы метода циклической редукции

$$x_j = a_j^{(l)} x_{j\text{-}2^l} + b_j^{(l)}; \quad j = 1,2,\ldots,n; l = 0,1,\ldots,\log_2 n;$$

$$a_j^{(l)} = a_j^{(l-1)}\ a_{j-2^{l-1}}^{(l-1)}, \qquad b_j^{(l)} = a_j^{(l-1)} b_{j\text{-}2^{l\text{-}1}}^{(l\text{-}1)} + b_j^{(l-1)},$$

$$a_j^{(0)} = a_j, b_j^{(0)} = b_j; j = 1,2,\ldots,n.$$

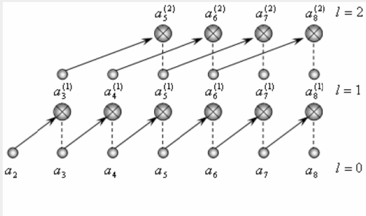
Если индекс ***j*** попадает вне диапазона изменения ***1 ≤ j ≤ n***, то правильный результат применения формул получается, если приравнять значение соответствующего элемента нулю. При $l = \log_2 n$ получаем явную формулу

$$\boxed{x_j = a_j^{(\log_2 n)} x_{j-n} + b_j^{(\log_2 n)} = b_j^{(\log_2 n)}, j = 1,\ldots,n.}$$

Суперкомпьютерный консорциум университетов России

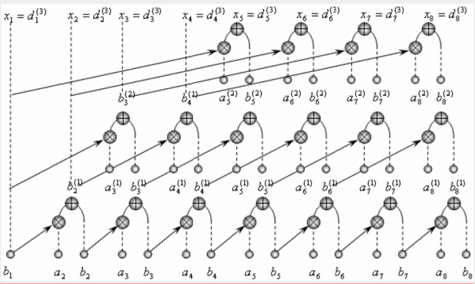
Метод циклической редукции

- Степень параллелизма метода циклической редукции изменяется примерно от ***n*** на начальном уровне и приблизительно до ***n/2*** на конечном уровне.



Суперкомпьютерный консорциум университетов России

Оценка ускорения



$$\boxed{S_p = \frac{T_1}{T_p} \approx \frac{(t_{add} + t_{mult})\, n}{(2\, t_{mult} + t_{add}) \log_2 n} \approx \frac{2\, n}{3 \log_2 n}}$$

Суперкомпьютерный консорциум университетов России

Заключение

- Представлены способы преобразования последовательных расчетов по рекуррентным формулам в алгоритмы с высокой степенью параллелизма (каскадная схема суммирования и циклическая редукция).
- Для вычисления суммы и частичных сумм элементов последовательности рассмотрены каскадная схема суммирования, алгоритм сдваивания, модифицированная каскадная схема суммирования.
- На примере каскадной схемы суммирования для вычисления частичных сумм элементов последовательности показано, что при организации одновременных и независимых вычислений объем арифметических операций может возрастать, а время выполнения алгоритма уменьшаться.
- Для линейной рекуррентной формулы применение циклической редукции позволяет получить явные расчетные формулы с высокой степенью параллелизма.

Суперкомпьютерный консорциум университетов России

Вопросы для обсуждения

- Приведите примеры использования рекуррентных соотношений в вычислительной математике.
- В чем заключается проблема распараллеливания рекуррентных соотношений?
- Что собой представляет диаграмма маршрутизации алгоритма и для чего она используется?
- Опишите алгоритм каскадной схемы суммирования.
- В чем заключается параллелизм каскадной схемы суммирования?
- За счет чего удается обеспечить равномерную работу ПЭ при выполнении каскадной схемы суммирования?
- Опишите алгоритм сдваивания для вычисления суммы элементов последовательности.
- Оцените количество операций сложения в алгоритме сдваивания при вычислении суммы ***n=2^q*** чисел.
- Опишите алгоритм модифицированной каскадной схемы суммирования элементов последовательности (МКСС).
- Каким ограничениям должно соответствовать число элементов суммируемой последовательности в МКСС?
- Дате оценку ускорения и эффективности МКСС.
- В чем заключается особенность МКСС по сравнению с алгоритмом сдваивания? За счет чего эта особенность проявляется?
- В чем заключается основная идея метода циклической редукции?
- Как применяется метод циклической редукции для распараллеливания линейной рекуррентной формулы первого порядка?
- После применения циклической редукции к линейным рекуррентным формулам удается ли получить явные формулы?
- Какова степень параллелизма полученного алгоритма?

Суперкомпьютерный консорциум университетов России

Темы заданий для самостоятельной работы

- Опишите алгоритм и приведите диаграмму маршрутизации для каскадной схемы суммирования ***n=16*** элементов последовательности.
- Составьте и протестируйте MPI-программу, реализующую каскадную схему суммирования ***n=2^q*** чисел.
- Опишите алгоритм и приведите диаграмму маршрутизации суммирования ***n=16*** чисел по алгоритму сдваивания.
- Составьте и протестируйте MPI-программу, реализующую суммирование ***n=2^q*** чисел по алгоритму сдваивания.
- Составьте и протестируйте MPI-программу, реализующую суммирование ***n=2^q*** (***n mod q = 0***) чисел по алгоритму модифицированной каскадной схемы суммирования.
- Опишите алгоритм и приведите диаграмму маршрутизации вычисления элементов последовательности $x_i=4x_{i-1}-2$, $i=1,\ldots,16$; $x_0=0$ с использованием циклической редукции.
- Составьте и протестируйте MPI-программу, реализующую вычисление элементов последовательности $x_i=-x_{i-1}+2$, $i=1,\ldots,2^q$; $x_0=0$ с использованием циклической редукции.

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.
- *Гергель В.П., Стронгин Р.Г.* Основы параллельных вычислений для многопроцессорных вычислительных машин. – Нижний Новгород: Изд-во ННГУ им. Н.И.Лобачевского, 2000.
- *Старченко А.В., Есаулов А.О.* Параллельные вычисления на многопроцессорных вычислительных системах. – Томск: Изд-во Том. ун-та, 2002. – 56 с.
- *Миллер Р., Боксер Л.* Последовательные и параллельные алгоритмы. – М.: БИНОМ. Лаборатория знаний, 2006.

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