СУПЕРКОМПЬЮТЕРНЫЙ КОНСОРЦИУМ УНИВЕРСИТЕТОВ РОССИИ

Проект

Создание системы подготовки высококвалифицированных кадров в области суперкомпьютерных технологий и специализированного программного обеспечения

Московский государственный университет им. М.В. Ломоносова

Нижегородский государственный университет им. Н.И. Лобачевского

- Национальный исследовательский университет -

Учебный курс

ВЫСОКОПРОИЗВОДИТЕЛЬНЫЕ ВЫЧИСЛЕНИЯ ДЛЯ МНОГОПРОЦЕССОРНЫХ МНОГОЯДЕРНЫХ СИСТЕМ

Лекция 2.

Моделирование и анализ параллельных вычислений

Гергель В.П., профессор, д.т.н.

Содержание

- Модель вычислений в виде графа "операции-операнды"
- Описание схемы параллельного выполнения алгоритма
- Определение времени выполнения параллельного алгоритма
- Показатели эффективности параллельного алгоритма
- Вычисление частных сумм последовательности числовых значений
  - Последовательный алгоритм суммирования
  - Каскадная схема суммирования
  - Модифицированная каскадная схема
  - Вычисление всех частных сумм
- Оценка максимально достижимого параллелизма
- Анализ масштабируемости параллельных вычислений

Моделирование и анализ параллельных вычислений

- При разработке параллельных алгоритмов решения сложных научно-технических задач принципиальным моментом является анализ эффективности использования параллелизма.
- Существует несколько подходов к анализу эффективности параллелизма:
  - Оценка получаемого ускорения процесса вычислений.
    - Формирование подобных оценок ускорения может осуществляться применительно к выбранному вычислительному алгоритму.
  - Построении оценок максимально возможного ускорения процесса решения задачи конкретного типа.
    - Оценка эффективности параллельного способа решения задачи.

Модель вычислений в виде графа "операции-операнды" ...

- Для описания существующих информационных зависимостей в выбираемых алгоритмах решения задач может быть использована модель в виде графа "операции-операнды".
  - Для уменьшения сложности излагаемого материала при построении модели будет предполагаться, что время выполнения любых вычислительных операций является одинаковым и равняется 1.
  - При построении моделей предполагается, что передача данных между вычислительными устройствами выполняется мгновенно без каких-либо затрат времени.

Модель вычислений в виде графа "операции-операнды" ...

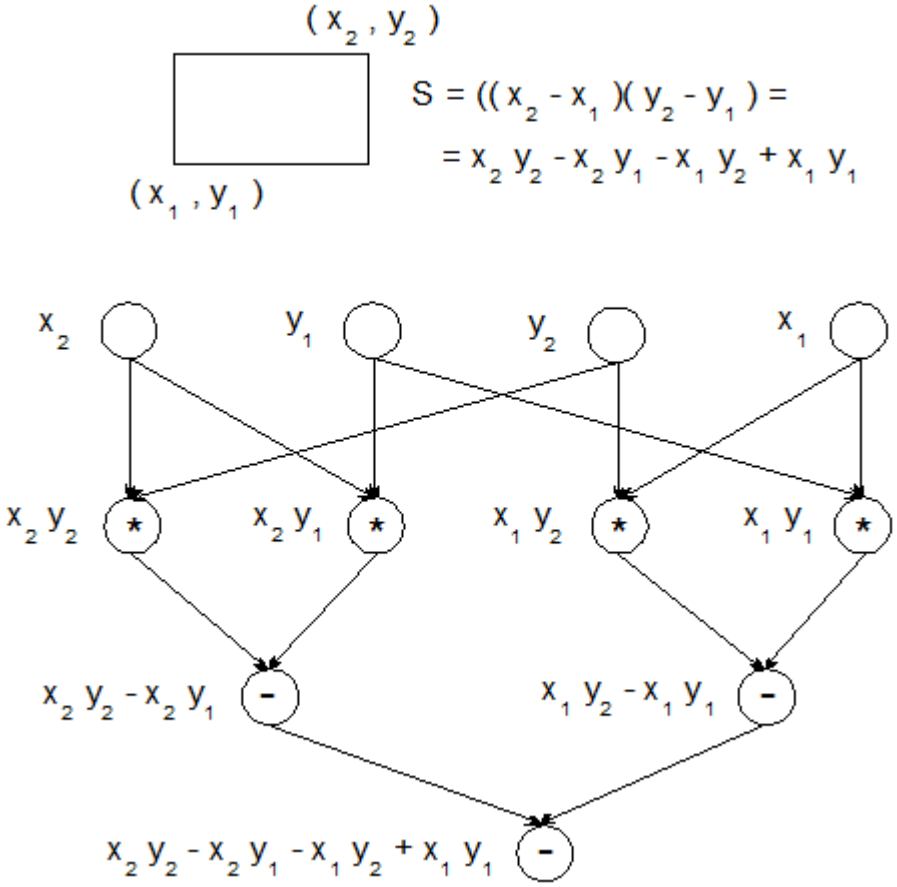
- Представим множество операций, выполняемых в исследуемом алгоритме решения вычислительной задачи, и существующие между операциями информационные зависимости в виде ациклического ориентированного графа

$$G = (V, R)$$

  - $V=\{1, \ldots, |V|\}$ есть множество вершин графа, представляющих выполняемые операции алгоритма
  - R есть множество дуг графа
    - дуга $r = (i, j)$ принадлежит графу только, если операция i использует результат выполнения операции j

Модель вычислений в виде графа "операции-операнды" ...

- Граф алгоритма вычисления площади прямоугольника, заданного координатами двух противолежащих углов.
- Разные схемы вычислений обладают различными возможностями для распараллеливания и, тем самым, при построении модели вычислений может быть поставлена задача выбора наиболее подходящей для параллельного исполнения вычислительной схемы алгоритма.



Модель вычислений в виде графа "операции-операнды"

- В рассматриваемой вычислительной модели алгоритма:
  - вершины без входных дуг могут использоваться для задания операций ввода,
  - вершины без выходных дуг – для операций вывода.
- Обозначим через $\overline{V}$ множество вершин графа без вершин ввода, а через $d(G)$ диаметр (длину максимального пути) графа.

Описание схемы параллельного выполнения алгоритма ...

- Операции алгоритма, между которыми нет пути в рамках выбранной схемы вычислений, могут быть выполнены параллельно.
- Для представленной схемы вычисления площади прямоугольника, параллельно могут быть реализованы сначала все операции умножения, а затем первые две операции вычитания.

Описание схемы параллельного выполнения алгоритма ...

- Возможный способ описания параллельного выполнения алгоритма может состоять в следующем:
  - Пусть p есть количество процессоров, используемых для выполнения алгоритма.
  - Тогда для параллельного выполнения вычислений необходимо задать множество (*расписание*), в котором для каждой операции i указывается номер используемого для выполнения операции процессора P_i и время начала выполнения операции t_i.

$$H_p = \{(i, P_i, t_i) : i \in V\}$$

Описание схемы параллельного выполнения алгоритма

- Для того, чтобы расписание было реализуемым, необходимо выполнение следующих требований при задании множества H_p:

$$\forall i, j \in V : t_i = t_j \Rightarrow P_i \neq P_j$$

  - т.е. один и тот же процессор не должен назначаться разным операциям в один и тот же момент времени,

$$\forall (i, j) \in R \Rightarrow t_j \geq t_i + 1$$

  - т.е. к назначаемому моменту выполнения операции все необходимые данные уже должны быть вычислены.

Определение времени выполнения параллельного алгоритма ...

- Вычислительная схема алгоритма совместно с расписанием H_p может рассматриваться как модель параллельного алгоритма $A_p(G, H_p)$, исполняемого с использованием p процессоров.

Определение времени выполнения параллельного алгоритма ...

- Время выполнения параллельного алгоритма определяется максимальным значением времени, используемым в расписании

$$T_p(G, H_p) = \max_{i \in V}(t_i + 1)$$

- Для выбранной схемы вычислений желательно использование расписания, обеспечивающего минимальное время исполнения алгоритма

$$T_p(G) = \min_{H_p} T_p(G, H_p)$$

- Уменьшение времени выполнения может быть обеспечено и путем подбора наилучшей вычислительной схемы

$$T_p = \min_{G} T_p(G)$$

Определение времени выполнения параллельного алгоритма ...

- Для анализа максимально возможного параллелизма можно определить оценку наиболее быстрого исполнения алгоритма

$$T_\infty = \min_{p \geq 1} T_p$$

- Оценку T_∞ можно рассматривать как минимально возможное время выполнения параллельного алгоритма при использовании неограниченного количества процессоров.
  - Концепция вычислительной системы с бесконечным количеством процессоров, обычно называемой *паракомпьютером*, широко используется при теоретическом анализе параллельных вычислений.

Определение времени выполнения параллельного алгоритма ...

- Оценка T_1 определяет время выполнения алгоритма при использовании одного процессора и представляет, тем самым, время выполнения последовательного варианта алгоритма решения задачи.

$$T_1(G) = \left|\overline{V}\right|$$

  - Построение подобной оценки является важной задачей при анализе параллельных алгоритмов, поскольку она необходима для определения эффекта использования параллелизма (ускорения времени решения задачи).

Определение времени выполнения параллельного алгоритма ...

- Если при определении оценки T_1 ограничиться рассмотрением только одного выбранного алгоритма решения задачи и использовать величину

$$T_1 = \min_G T_1(G)$$

то получаемые при использовании такой оценки показатели ускорения будут характеризовать эффективность распараллеливания **выбранного алгоритма**.

- Для оценки эффективности параллельного решения исследуемой задачи вычислительной математики время последовательного решения следует определять с **учетом различных последовательных алгоритмов**, т.е. использовать величину

$$T_1^* = \min T_1$$

Определение времени выполнения параллельного алгоритма ...

- **Теорема 1.** Минимально возможное время выполнения параллельного алгоритма определяется длиной максимального пути вычислительной схемы алгоритма, т.е.

$$T_\infty(G) = d(G)$$

- **Теорема 2.** Пусть для некоторой вершины вывода в вычислительной схеме алгоритма существует путь из каждой вершины ввода. Кроме того, пусть входная степень вершин схемы (количество входящих дуг) не превышает 2. Тогда минимально возможное время выполнения параллельного алгоритма ограничено снизу значением

$$T_\infty(G) = \log_2 n$$

где n есть количество вершин ввода в схеме алгоритма.

Определение времени выполнения параллельного алгоритма ...

- **Теорема 3.** При уменьшении числа используемых процессоров время выполнения алгоритма увеличивается пропорционально величине уменьшения количества процессоров, т.е.

$$\forall q = cp, \quad 0 < c < 1 \Rightarrow T_p \le cT_q$$

- **Теорема 4.** Для любого количества используемых процессоров справедлива следующая верхняя оценка для времени выполнения параллельного алгоритма

$$\forall p \Rightarrow T_p < T_\infty + T_1 / p$$

Определение времени выполнения параллельного алгоритма ...

- **Теорема 5.** Времени выполнения алгоритма, которое сопоставимо с минимально возможным временем T_∞, можно достичь при количестве процессоров порядка $p \sim T_1 / T_\infty$, а именно,

$$p \geq T_1 / T_\infty \Rightarrow T_p \leq 2T_\infty$$

При меньшем количестве процессоров время выполнения алгоритма не может превышать более, чем в 2 раза, наилучшее время вычислений при имеющемся числе процессоров, т.е.

$$p < T_1 / T_\infty \Rightarrow \frac{T_1}{p} \leq T_p \leq 2\frac{T_1}{p}$$

Определение времени выполнения параллельного алгоритма ...

- Приведенные утверждения позволяют дать следующие рекомендации по правилам формирования параллельных алгоритмов:
  - при выборе вычислительной схемы алгоритма должен использоваться граф с минимально возможным диаметром (см. теорему 1);
  - для параллельного выполнения целесообразное количество процессоров определяется величиной $p \sim T_1 / T_\infty$ (см. теорему 5);
  - время выполнения параллельного алгоритма ограничивается сверху величинами, приведенными в теоремах 4 и 5.

Определение времени выполнения параллельного алгоритма ...

- **Доказательство теоремы 4.**
  - Пусть H_∞ есть расписание для достижения минимально возможного времени выполнения T_∞.
  - Для каждой итерации τ, $0 \le \tau \le T_\infty$ выполнения расписания H_∞ обозначим через n_τ количество операций, выполняемых в ходе итерации τ. Расписание выполнения алгоритма с использованием p процессоров может быть построено следующим образом.
    - Выполнение алгоритма разделим на T_∞ шагов;
    - На каждом шаге τ следует выполнить все n_τ операций, которые выполнялись на итерации τ расписания H_∞.
    - Выполнение этих операций может быть выполнено не более, чем за $\lceil n_\tau / p \rceil$ итераций при использовании p процессоров.
    - Как результат, время выполнения алгоритма может быть оценено следующим образом

$$T_p = \sum_{\tau=1}^{T_\infty} \left\lceil \frac{n_\tau}{p} \right\rceil < \sum_{\tau=1}^{T_\infty} \left(\frac{n_\tau}{p} + 1 \right) = \frac{T_1}{p} + T_\infty$$

Определение времени выполнения параллельного алгоритма

- Доказательство теоремы дает практический способ построения расписания параллельного алгоритма.
  - Первоначально может быть построено расписание без учета ограниченности числа используемых процессоров (расписание для паракомпьютера).
  - Затем, согласно схеме вывода теоремы, может быть построено расписание для конкретного количества процессоров.

Показатели эффективности параллельного алгоритма ...

- ***Ускорение (speedup)***, получаемое при использовании параллельного алгоритма для процессоров, по сравнению с последовательным вариантом выполнения вычислений определяется величиной

$$S_p(n) = T_1(n) / T_p(n)$$

- ***Эффективность (efficiency)*** использования параллельным алгоритмом процессоров при решении задачи определяется соотношением

$$E_p(n) = T_1(n) / (pT_p(n)) = S_p(n) / p$$

Показатели эффективности параллельного алгоритма ...

- Из приведенных соотношений можно показать, что в наилучшем случае $S_p(n) = p$ и $E_p(n) = 1$.
- При практическом применении данных показателей для оценки эффективности параллельных вычислений следует учитывать следующих два важных момента:
  - При определенных обстоятельствах ускорение может оказаться больше числа используемых процессоров $S_p(n) > p$ - в этом случае говорят о существовании *сверхлинейного* (*superlinear*) ускорения.
  - При внимательном рассмотрении можно обратить внимание, что попытки повышения качества параллельных вычислений по одному из показателей (ускорению или эффективности) может привести к ухудшению ситуации по другому показателю, ибо показатели качества параллельных вычислений являются противоречивыми.

Показатели эффективности параллельного алгоритма

- При выборе надлежащего параллельного способа решения задачи может оказаться полезной оценка ***стоимости*** (***cost***) вычислений, определяемой как произведение времени параллельного решения задачи и числа используемых процессоров

$$C_p = pT_p$$

- В этой связи можно определить понятие ***стоимостно-оптимального*** (***cost-optimal***) параллельного алгоритма как метода, стоимость которого является пропорциональной времени выполнения наилучшего последовательного алгоритма.

Вычисление частных сумм последовательности числовых значений

- Рассмотрим для демонстрации ряда проблем, возникающих при разработке параллельных методов вычислений, сравнительно простую *задачу нахождения частных сумм* последовательности числовых значений

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

где есть количество суммируемых значений (данная задача известна также под названием *prefix sum problem*).

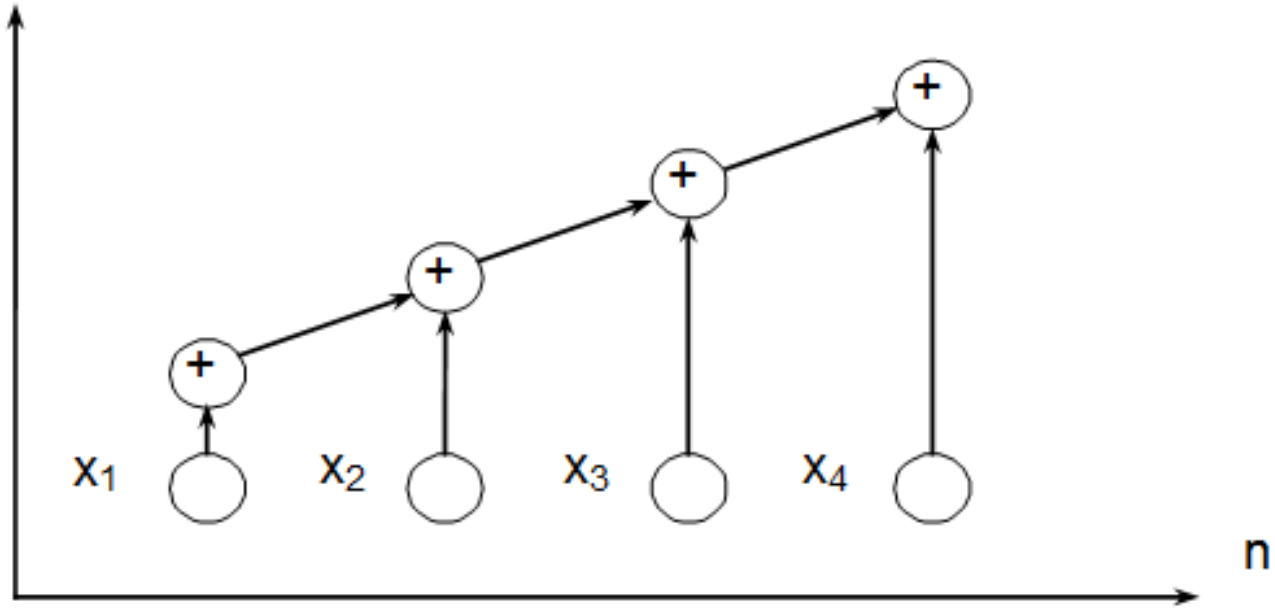
- Изучение возможных параллельных методов решения данной задачи начнем с еще более простого варианта ее постановки – с *задачи вычисления общей суммы* имеющегося набора значений (в таком виде задача суммирования является частным случаем общей *задачи редукции*)

$$S = \sum_{i=1}^{n} x_i$$

Последовательный алгоритм суммирования

- Традиционный алгоритм для решения задачи суммирования

$$S = 0,$$
$$S = S + x_1, \ldots$$



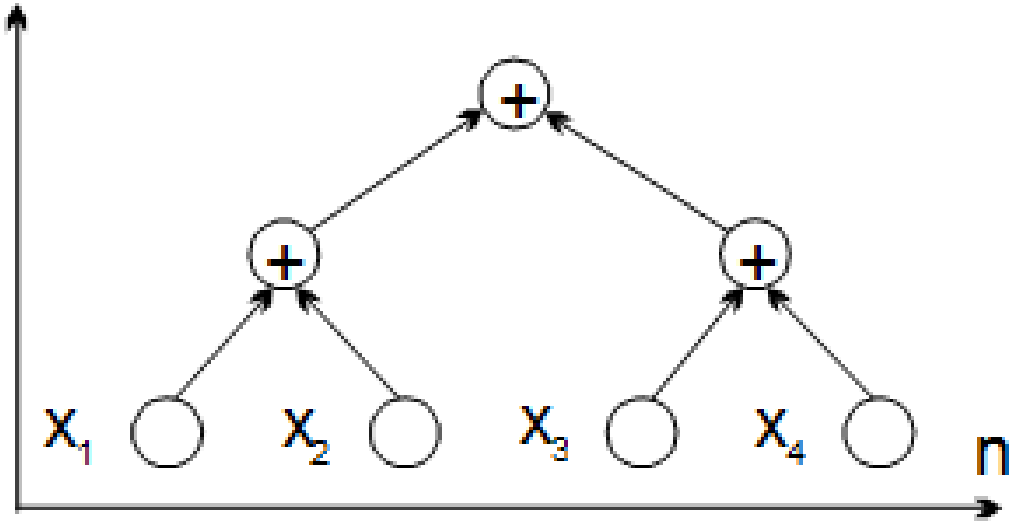
- Как можно заметить, данный "стандартный" алгоритм суммирования допускает только строго последовательное исполнение и не может быть распараллелен.

Каскадная схема суммирования ...

- **Параллелизм алгоритма суммирования становится возможным только при ином способе построения процесса вычислений**, основанном на использовании ассоциативности операции сложения.
- Получаемый новый вариант суммирования (известный в литературе как *каскадная схема*) состоит в следующем:
  - на первой итерации каскадной схемы все исходные данные разбиваются на пары, и для каждой пары вычисляется сумма их значений,
  - далее все полученные суммы также разбиваются на пары, и снова выполняется суммирование значений пар и т.д.

Каскадная схема суммирования ...

- Вычислительная схема каскадного суммирования



Каскадная схема суммирования ...

- Как нетрудно оценить, количество итераций каскадной схемы оказывается равным величине

$$k = \log_2 n$$

- Общее количество операций суммирования совпадает с количеством операций последовательного варианта алгоритма суммирования.

$$K_{посл} = n/2 + n/4 + \ldots + 1 = n - 1$$

- При параллельном исполнении отдельных итераций каскадной схемы общее количество параллельных операций суммирования является равным

$$K_{пар} = \log_2 n$$

Каскадная схема суммирования

- Поскольку считается, что время выполнения любых вычислительных операций является одинаковым и единичным, то $T_1 = K_{посл}$, $T_p = K_{пар}$, поэтому показатели ускорения и эффективности каскадной схемы алгоритма суммирования можно оценить как

$$S_P = T_1 / T_P = (n-1)/\log_2 n,$$

$$E_p = T_1 / pT_p = (n-1)/(p\log_2 n) = (n-1)/((n/2)\log_2 n),$$

где p есть необходимое для выполнения каскадной схемы количество процессоров.

- **Эффективность использования процессоров уменьшается при увеличении количества суммируемых значений**

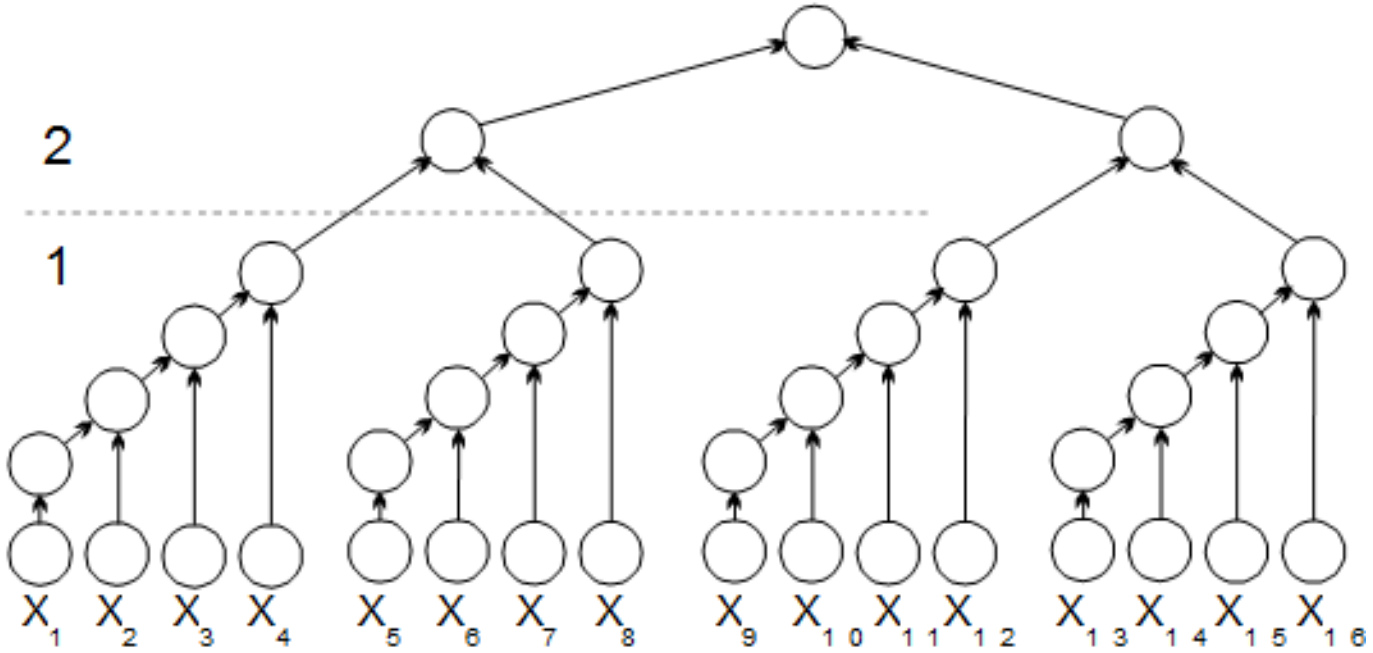
$$\lim E_P \to 0 \quad при \quad n \to \infty$$

Модифицированная каскадная схема ...

- Получение асимптотически ненулевой эффективности может быть обеспечено, например, при использовании модифицированной каскадной схемы.
- Для упрощения построения оценок можно предположить $n = 2^k$, $k = 2^s$. Тогда в новом варианте каскадной схемы все проводимые вычисления подразделяются на два последовательно выполняемых этапа суммирования:
  - на первом этапе вычислений все суммируемые значения подразделяются на $(n/\log_2 n)$ групп, в каждой из которых содержится элементов; далее для каждой группы вычисляется сумма значений при помощи последовательного алгоритма суммирования; вычисления в каждой группе могут выполняться независимо друг от друга;
  - на втором этапе для полученных $(n/\log_2 n)$ сумм отдельных групп применяется обычная каскадная схема.

Модифицированная каскадная схема ...

- Вычислительная схема



Модифицированная каскадная схема ...

- Тогда для выполнения первого этапа требуется выполнение $\log_2 n$ параллельных операций при использовании $p_1 = (n / \log_2 n)$ процессоров.
- Для выполнения второго этапа необходимо

$$\log_2 (n / \log_2 n) \le \log_2 n$$

параллельных операций для $p_2 = (n / \log_2 n) / 2$ процессоров.

- Как результат, данный способ суммирования характеризуется следующими показателями:

$$T_P = 2\log_2 n \quad p = (n / \log_2 n)$$

$$S_P = T_1 / T_P = (n-1) / 2\log_2 n,$$

$$E_p = T_1 / pT_p = (n-1) / (2(n / \log_2 n)\log_2 n) = (n-1) / 2n.$$

Модифицированная каскадная схема

- Сравнивая данные оценки с показателями обычной каскадной схемы, можно отметить, что ускорение для предложенного параллельного алгоритма уменьшилось в 2 раза, однако для эффективности нового метода суммирования можно получить асимптотически ненулевую оценку снизу

$$E_P = (n-1)/2n \geq 0.25,$$

$$\lim E_P \to 0.5 \quad при \quad n \to \infty$$

- Кроме того, необходимо подчеркнуть, что, в отличие от обычной каскадной схемы, модифицированный каскадный алгоритм является *стоимостно-оптимальным*, поскольку стоимость вычислений в этом случае является пропорциональной времени выполнения последовательного алгоритма.

$$C_p = pT_p = (n/\log_2 n)(2\log_2 n)$$

Вычисление всех частных сумм ...

- Вычисление всех частных сумм на скалярном компьютере может быть получено при помощи обычного последовательного алгоритма суммирования при том же количестве операций (!)

$$T_1 = n$$

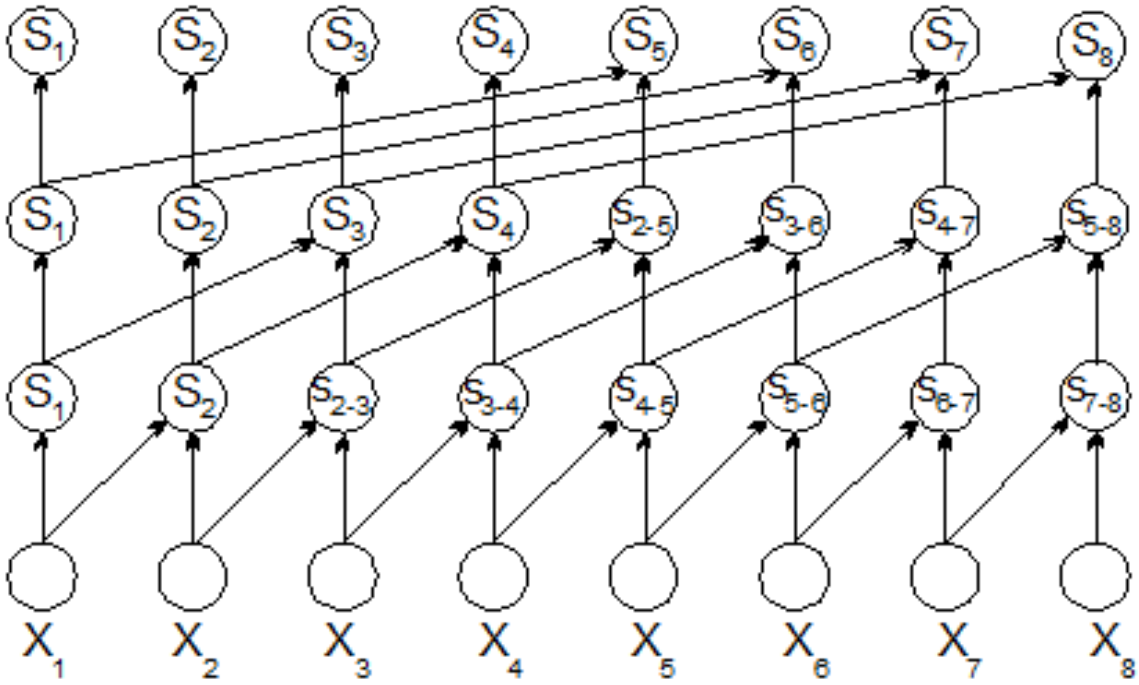
- При параллельном исполнении применение каскадной схемы в явном виде не приводит к желаемым результатам; достижение эффективного распараллеливания требует привлечения новых подходов (может быть, даже не имеющих аналогов при последовательном программировании) для разработки новых параллельно-ориентированных алгоритмов решения задач.

Вычисление всех частных сумм …

- Для рассматриваемой задачи нахождения всех частных сумм, алгоритм, обеспечивающий получение результатов за ($\log_2 n$) параллельных операций, может состоять в следующем:
  - перед началом вычислений создается копия S вектора суммируемых значений ($S = x$);
  - далее на каждой итерации суммирования i, ($1 \le i \le \log_2 n$) формируется вспомогательный вектор Q путем сдвига вправо вектора S на 2^{i-1} позиций (освобождающиеся при сдвиге позиции слева устанавливаются в нулевые значения);
  - итерация алгоритма завершается параллельной операцией суммирования векторов S и Q.

Вычисление всех частных сумм ...

- Схема параллельного алгоритма вычисления всех частных сумм



Вычисление всех частных сумм ...

- Всего параллельный алгоритм выполняется за ($\log_2 n$) параллельных операций сложения. На каждой итерации алгоритма параллельно выполняются n скалярных операций сложения и, таким образом, общее количество выполняемых скалярных операций определяется величиной

$$K_{пар} = n \log_2 n$$

- **Параллельный алгоритм содержит большее количество операций по сравнению с последовательным способом суммирования.**
- Необходимое количество процессоров определяется количеством суммируемых значений ($p = n$).

Вычисление всех частных сумм

- Показатели ускорения и эффективности параллельного алгоритма вычисления всех частных сумм оцениваются следующим образом

$$S_P = T_1 / T_P = n / \log_2 n,$$

$$E_P = T_1 / pT_P = n /(p \log_2 n) = n /(n \log_2 n) = 1 / \log_2 n$$

- Как следует из построенных оценок, *эффективность алгоритма также уменьшается* при увеличении числа суммируемых значений и при необходимости повышения величины этого показателя может оказаться полезной модификация алгоритма, как и в случае с обычной каскадной схемой.

Оценка максимально достижимого параллелизма ...

- Оценка качества параллельных вычислений предполагает знание *наилучших* (*максимально достижимых*) значений показателей ускорения и эффективности
- Получение идеальных величин $S_p=p$ для ускорения и $E_p=1$ для эффективности может быть обеспечено не для всех вычислительно трудоемких задач.
  - Для рассматриваемого учебного примера в предыдущем пункте минимально достижимое время параллельного вычисления суммы числовых значений составляет $log_2 n$.

Оценка максимально достижимого параллелизма ...

- **Закон Амдаля.**
  - Достижению максимального ускорения может препятствовать существование в выполняемых вычислениях последовательных расчетов, которые не могут быть распараллелены.
    - Пусть f есть *доля последовательных вычислений* в применяемом алгоритме обработки данных, тогда в соответст-вии с *законом Амдаля* (*Amdahl*) ускорение процесса вычислений при использовании p процессоров ограничивается величиной

$$S_p \le \frac{1}{f + (1-f)/p} \le S^* = \frac{1}{f}$$

Оценка максимально достижимого параллелизма ...

- **Закон Амдаля.**
  - Закон Амдаля характеризует одну из самых серьезных проблем в области параллельного программирования.
  - Например, при наличии всего 10% последовательных команд в выполняемых вычислениях, эффект использования параллелизма не может превышать 10-кратного ускорения обработки данных.

Оценка максимально достижимого параллелизма ...

- **Закон Густавсона - Барсиса.**
  - Оценим максимально достижимое ускорение исходя из имеющейся доли последовательных расчетов в выполняемых параллельных вычислениях:

$$g = \frac{\tau(n)}{\tau(n) + \pi(n)/p}$$

где $\tau(n)$ и $\pi(n)$ есть времена последовательной и параллельной частей выполняемых вычислений соответственно, т.е.

$$T_1 = \tau(n) + \pi(n)$$

$$T_p = \tau(n) + \pi(n)/p$$

Оценка максимально достижимого параллелизма ...

- **Закон Густавсона - Барсиса.**
  - С учетом введенной величины g можно получить

$$\tau(n)=g\cdot(\tau(n)+\pi(n)/p),\quad \pi(n)=(1-g)p\cdot(\tau(n)+\pi(n)/p),$$

  - что позволяет построить оценку для ускорения

$$S_p=\frac{T_1}{T_p}=\frac{\tau(n)+\pi(n)}{\tau(n)+\pi(n)/p}=\frac{(\tau(n)+\pi(n)/p)(g+(1-g)p)}{\tau(n)+\pi(n)/p}$$

  - которая после упрощения приводится к виду *закона Густавсона-Барсиса* (*Gustafson-Barsis's law*)

$$S_p=g+(1-g)p=p+(1-p)g$$

Оценка максимально достижимого параллелизма

- **Закон Густавсона - Барсиса.**
  - Оценку ускорения, получаемую в соответствии с законом Густавсона-Барсиса, еще называют *ускорением маситабирования* (*scaled speedup*), поскольку данная характеристика может показать, **насколько эффективно могут быть организованы параллельные вычисления при увеличении сложности решаемых задач**.

Анализ масштабируемости параллельных вычислений ...

- Целью применения параллельных вычислений во многих случаях является не только уменьшение времени выполнения расчетов, но и обеспечение возможности решения более сложных вариантов решаемых задач.
- Способность параллельного алгоритма эффективно использовать процессоры при повышении сложности вычислений является важной характеристикой выполняемых расчетов.
- Параллельный алгоритм называют *масштабируемым* (*scalable*), если при росте числа процессоров он обеспечивает увеличение ускорения при сохранении постоянного уровня эффективности использования процессоров.

Анализ масштабируемости параллельных вычислений ...

- Оценим *накладные расходы* (*total overhead*), которые имеют место при выполнении параллельного алгоритма

$$T_0 = pT_p - T_1$$

- Используя введенное обозначение, можно получить новые выражения для времени параллельного решения задачи и соответствующего ускорения:

$$T_p = \frac{T_1 + T_0}{p} \qquad S_p = \frac{T_1}{T_p} = \frac{pT_1}{T_1 + T_0}$$

- Используя полученные соотношения, эффективность использования процессоров можно выразить как

$$E_p = \frac{S_p}{p} = \frac{T_1}{T_1 + T_0} = \frac{1}{1 + T_0 / T_1}$$

Анализ масштабируемости параллельных вычислений ...

- Последнее выражение показывает, что, если сложность решаемой задачи является фиксированной (T_1=*const*), то при росте числа процессоров эффективность, как правило, будет убывать за счет роста накладных расходов T_0.
- При фиксации числа процессоров эффективность использования процессоров можно улучшить путем повышения сложности решаемой задачи T_1 (предполагается, что при росте параметра сложности n накладные расходы T_0 увеличиваются медленнее, чем объем вычислений T_1).
  - Как результат, при увеличении числа процессоров в большинстве случаев можно обеспечить определенный уровень эффективности при помощи соответствующего повышения сложности решаемых задач.

Анализ масштабируемости параллельных вычислений ...

- Пусть $E=const$ есть желаемый уровень эффективности выполняемых вычислений. Из выражения для эффективности можно получить

$$\frac{T_0}{T_1} = \frac{1-E}{E} \text{ или } T_1 = KT_0,\ K = E/(1-E)\ .$$

- Порождаемую последним соотношением зависимость $n=F(p)$ между сложностью решаемой задачи и числом процессоров обычно называют *функцией изоэффективности* (*isoefficiency function*)

Анализ масштабируемости параллельных вычислений

- Покажем в качестве иллюстрации вывод функции изоэффективности для учебного примера суммирования числовых значений. В этом случае

$$T_0 = pT_p - T_1 = p((n/p) + \log_2 p) - n = p\log_2 p$$

и функция изоэффективности принимает вид

$$n = Kp\log_2 p$$

- Как результат, например, при числе процессоров p=16 для обеспечения уровня эффективности E=0.5 (т.е. K=1) количество суммируемых значений должно быть не менее n=64.
- Или же, при увеличении числа процессоров с p до q (q>p) для обеспечения пропорционального роста ускорения $(S_q/S_p)=(q/p)$ необходимо увеличить число суммируемых значений n в $(q\log_2 q)/(p\log_2 p)$ раз.

Заключение…

- В разделе описывается модель вычислений в виде графа "операции-операнды", которая может использоваться для описания существующих информационных зависимостей в выбираемых алгоритмах решения задач.
- Рассматривается понятие *паракомпьютера* как параллельной системы с неограниченным количеством процессоров.
- Для оценки оптимальности разрабатываемых методов параллельных вычислений в разделе приводятся основные показатели качества - ускорение (*speedup*), эффективность (*efficiency*), стоимость (*cost*) вычислений.

Заключение

- Для демонстрации применимости рассмотренных моделей и методов анализа параллельных алгоритмов в разделе рассматривается задача нахождения частных сумм последовательности числовых значений
- Рассматривается вопрос построения оценок максимально достижимых значений показателей эффективности. Для получения таких оценок может быть использован *закон Амдаля* (*Amdahl*) и *закон Густавсона-Барсиса* (*Gustafson-Barsis's law*)
- Вводится понятие *функции изоэффективности* (*isoefficiency function*)
- Получение описанных оценок иллюстрируется на примерах

Вопросы для обсуждения …

- Как определяется модель "операция - операнды"?
- Как определяется расписание для распределения вычислений между процессорами?
- Как определяется время выполнения параллельного алгоритма?
- Какое расписание является оптимальным?
- Как определить минимально возможное время решения задачи?
- Какие оценки следует использовать в качестве характеристики времени последовательного решения задачи?
- Как определить минимально возможное время параллельного решения задачи по графу "операнды – операции"?
- Какие зависимости могут быть получены для времени параллельного решения задачи при увеличении или уменьшения числа используемых процессоров?
- При каком числе процессоров могут быть получены времена выполнения параллельного алгоритма, сопоставимые по порядку с оценками минимально возможного времени решения задачи?

Вопросы для обсуждения

- Как определяются понятия ускорения и эффективности?
- Возможно ли достижений сверхлинейного ускорения?
- В чем состоит противоречивость показателей ускорения и эффективности?
- Как определяется понятие стоимости вычислений?
- Как формулируется закон Амдаля? Какой аспект параллельных вычислений позволяет учесть данный закон?
- Какие предположения используются для обоснования закона Густавсона-Барсиса?
- Какой алгоритм является масштабируемым? Приведите примеры методов с разным уровнем масштабируемости.

Темы заданий для самостоятельной работы…

- Разработайте модель и выполните оценку показателей ускорения и эффективности параллельных вычислений:
  - для задачи скалярного произведения двух векторов

$$y = \sum_{i=1}^{N} a_i b_i$$

  - для задачи поиска максимального и минимального значений для заданного набора числовых данных

$$y_{\min} = \min_{i \le i \le N} a_i, \; y_{\max} = \max_{i \le i \le N} a_i$$

  - для задачи нахождения среднего значения для заданного набора числовых данных

$$y = \frac{1}{N} \sum_{i=1}^{N} a_i$$

Темы заданий для самостоятельной работы

- Выполните в соответствии с законом Амдаля оценку максимально достижимого ускорения для поставленных задач.
- Выполните оценку ускорения масштабирования для поставленных задач.
- Выполните построение функций изоэффективности для поставленных задач.
- (*) Разработайте модель и выполните полный анализ эффективности параллельных вычислений (ускорение, эффективность, максимально достижимое ускорение, ускорение масштабирования, функция изоэффективности) для задачи умножения матрицы на вектор.

Обзор литературы

- Дополнительная информация по моделированию и анализу параллельных вычислений может быть получена, например, в
  - Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2002.
  - Гергель В.П. Теория и практика параллельных вычислений. – М.: Интернет-Университет Информационных технологий; БИНОМ. Лаборатория знаний, 2007.
  - Bertsekas D.P., Tsitsiklis J.N. Parallel and Distribution Computation. Numerical Methods. – Prentice Hall, Englewood Cliffs, New Jersey, 1989
  - Kumar V., Grama A., Gupta A., Karypis G. Introduction to Parallel Computing , Inc. 1994 (2th edition, 2003)
  - Quinn M.J. Parallel Programming in C with MPI and OpenMP. – New York, NY: McGraw-Hill, 2004

Обзор литературы

- Рассмотрение учебной задачи суммирования последовательности числовых значений было выполнено в
  - Bertsekas D.P., Tsitsiklis J.N. Parallel and Distribution Computation. Numerical Methods. – Prentice Hall, Englewood Cliffs, New Jersey, 1989
- Впервые закон Амдаля был изложен в работе
  - Amdahl, G. Validity of the single processor approach to achieving large scale computing capabilities. In AFIPS Conference Proceedings, Vol. 30, 1967, pp. 461-485, Washington, D.C.: Thompson Books.
- Закон Густавсона-Барсиса был опубликован в работе
  - Gustavson, J.L. Reevaluating Amdahl's law. Communications of the ACM. 31 (5). 1988, pp.532-533.
- Понятие функции изоэффективности было предложено в работе
  - Grama, A.Y., Gupta, A. and Kumar, V. Isoefficiency: Measuring the scalability of parallel algorithms and architectures. IEEE Parallel and Distributed technology. 1 (3). 1993, pp. 12-21.
- Систематическое изложение вопросов моделирования и анализа параллельных вычислений приводится в
  - Zomaya, A.Y. (Ed.) Parallel and Distributed Computing Handbook. - McGraw-Hill, 1996.

Следующая тема

- Принципы разработки параллельных методов

О проекте

Целью проекта является создание национальной системы подготовки высококвалифицированных кадров в области суперкомпьютерных технологий и специализированного программного обеспечения.

Задачами по проекту являются:

Задача 1. Создание сети научно-образовательных центров суперкомпьютерных технологий (НОЦ СКТ).

Задача 2. Разработка учебно-методического обеспечения системы подготовки, переподготовки и повышения квалификации кадров в области суперкомпьютерных технологий.

Задача 3. Реализация образовательных программ подготовки, переподготовки и повышения квалификации кадров в области суперкомпьютерных технологий.

Задача 4. Развитие интеграции фундаментальных и прикладных исследований и образования в области суперкомпьютерных технологий. Обеспечение взаимодействия с РАН, промышленностью, бизнесом.

Задача 5. Расширение международного сотрудничества в создании системы суперкомпьютерного образования.

Задача 6. Разработка и реализации системы информационного обеспечения общества о достижениях в области суперкомпьютерных технологий.

См. http://www.hpc-education.ru