3. Моделирование и анализ параллельных вычислений. Модель вычислений в виде графа «операции-операнды». Описание схемы параллельного выполнения алгоритма. Определение времени выполнения параллельного алгоритма. Показатели эффективности параллельного алгоритма.

При разработке параллельных алгоритмов решения сложных научно-технических задач принципиальным моментом является анализ эффективности использования параллелизма, состоящий обычно в оценке получаемого ускорения процесса вычислений (сокращения времени решения задачи). Формирование подобных оценок ускорения может осуществляться применительно к выбранному вычислительному алгоритму (оценка эффективности распараллеливания конкретного алгоритма). Другой важный подход состоит в построении оценок максимально возможного ускорения процесса решения задачи конкретного типа (оценка эффективности параллельного способа решения задачи).

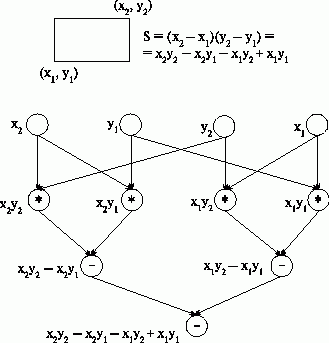
В данной лекции рассматривается модель вычислений в виде графа "операции – операнды", которая может использоваться для описания существующих информационных зависимостей в выбираемых алгоритмах решения задач, и приводятся оценки эффективности максимально возможного параллелизма, которые могут быть получены в результате анализа имеющихся моделей вычислений. Примеры применения излагаемого теоретического материала приводятся в лекциях 6 – 11 настоящего учебного пособия при изучении параллельных методов решения ряда сложных вычислительно трудоемких задач.

**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" (см., например, [2, 22]). Для уменьшения сложности излагаемого материала при построении модели будет предполагаться, что время выполнения любых вычислительных операций является одинаковым и равняется 1 (в тех или иных единицах измерения). Кроме того, принимается, что передача данных между вычислительными устройствами выполняется мгновенно без каких-либо затрат времени (что может быть справедливо, например, при наличии общей разделяемой памяти в параллельной вычислительной системе). Анализ коммуникационной трудоемкости параллельных алгоритмов приводится в следующей лекции.

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

G = (V, R),



Пример вычислительной модели алгоритма в виде графа "операции – операнды"

где V = {1,...,|V|} есть множество вершин графа, представляющих выполняемые операции алгоритма, а R есть множество дуг графа (при этом дуга r = (i, j) принадлежит графу только в том случае, если операция j использует результат выполнения операции i ). Для примера на рис. 2.1 показан граф алгоритма вычисления площади прямоугольника, заданного координатами двух противолежащих углов. Как можно заметить по приведенному примеру, для выполнения выбранного алгоритма решения задачи могут быть использованы разные схемы вычислений и построены соответственно разные вычислительные модели. Как будет показано далее, 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.

В рассматриваемой вычислительной модели алгоритма вершины без входных дуг могут использоваться для задания операций ввода, а вершины без выходных дуг – для операций вывода. Обозначим через 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 вычислительной схемы на рис. выше, например, параллельно могут быть реализованы сначала все операции умножения, а затем первые две операции вычитания). 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* )

H_{p} = \{ (i,P_{i},t_{i}):i\in V\} ,

в котором для каждой операции i\in V указывается номер используемого для выполнения операции процессора Pi и время начала выполнения операции ti. Для того чтобы *расписание* было реализуемым, необходимо выполнение следующих требований при задании множества Hp:

1. \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;
2. \forall(i,j) \in R \Rightaroow t_j \ge 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 G совместно с *расписанием* Hp может рассматриваться как модель параллельного алгоритма Ap(G,Hp), исполняемого с использованием p процессоров. Время выполнения параллельного алгоритма определяется максимальным значением времени, применяемым 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).

Оценки Tp(G,Hp), Tp(G) и Tp могут быть применены в качестве показателей времени выполнения параллельного алгоритма. Кроме того, для анализа максимально возможного параллелизма можно определить оценку наиболее быстрого исполнения алгоритма

T_{\infty} = \min_{p \ge 1} T_p.

Оценку T_{\infty } 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 применяется при теоретическом анализе параллельных вычислений).

Оценка T1 определяет *время выполнения* алгоритма при использовании одного процессора и представляет, тем самым, *время выполнения* последовательного варианта алгоритма решения задачи. Построение подобной оценки является важной задачей при анализе параллельных алгоритмов, поскольку она необходима для определения эффекта использования параллелизма (ускорения времени решения задачи). Очевидно, что

T_1 (G) = |\overline{V}|,

где |\overline{V}|, напомним, есть количество вершин вычислительной схемы без вершин ввода. Важно отметить, что если при определении оценки ограничиться рассмотрением только одного выбранного алгоритма решения задачи и использовать величину

T_1 = \min_{G} T_1 (G),

то получаемые при такой оценке показатели ускорения будут характеризовать *эффективность* распараллеливания выбранного алгоритма. Для оценки эффективности параллельного решения исследуемой вычислительной задачи время последовательного решения следует определять с учетом различных последовательных алгоритмов, т.е. использовать величину

T_1^* = \min T_1,

где операция минимума берется по множеству всех возможных последовательных алгоритмов решения данной задачи.

Приведем без доказательства теоретические положения, характеризующие свойства оценок времени выполнения параллельного алгоритма (см. [[22](https://intuit.ru/studies/courses/1156/190/literature#literature.22)]).

**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.

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

\forall q=cp,\ 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.

**Теорема 5**. Времени выполнения алгоритма, которое сопоставимо с минимально возможным временем T_{\infty }, можно достичь при количестве процессоров порядка p \sim T_{1}/T_{\infty }, а именно,

p\ge T_{1}/T_{\infty }\Rightarrow T_{p}\le 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} \le T_p \le 2 \frac{T_1}{p}.

Приведенные утверждения позволяют дать следующие рекомендации по правилам формирования параллельных алгоритмов:

1. при выборе вычислительной схемы алгоритма должен использоваться граф с минимально возможным диаметром (см. теорему 1);
2. для параллельного выполнения целесообразное количество процессоров определяется величиной p \sim T_{1}/T_{\infty } (см. теорему 5);
3. *время выполнения* параллельного алгоритма ограничивается сверху величинами, приведенными в теоремах 4 и 5.

Для вывода рекомендаций по формированию расписания по параллельному выполнению алгоритма приведем доказательство теоремы 4.

**Доказательство теоремы 4**. Пусть H_{\infty } есть *расписание* для достижения минимально возможного времени выполнения T_{\infty }. Для каждой итерации \tau ,\ 0<\tau <T_{\infty }, выполнения расписания H_{\infty } обозначим через n_{\tau } количество операций, выполняемых в ходе итерации \tau. *Расписание* выполнения алгоритма с использованием p процессоров может быть построено следующим образом. Выполнение алгоритма разделим на T_{\infty } шагов; на каждом шаге \tau следует выполнить все n_{\tau } операций, которые выполнялись на итерации \tau расписания H_{\infty }. Эти операции могут быть выполнены не более чем за \lceil n_{\tau }/p\rceil итераций при использовании p процессоров. Как результат, время выполнения алгоритма Tp 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} .

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 p процессоров, по сравнению с последовательным вариантом выполнения вычислений определяется величиной

Sp(n)=T1(n)/Tp(n),

т.е. как отношение времени решения задач на скалярной ЭВМ к времени выполнения параллельного алгоритма (величина n применяется для параметризации вычислительной сложности решаемой задачи и может пониматься, например, как количество входных данных задачи).

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

Ep(n)=T1(n)/(pTp(n))=Sp(n)/p

(величина эффективности определяет среднюю долю времени выполнения алгоритма, в течение которой процессоры реально задействованы для решения задачи).

Из приведенных соотношений можно показать, что в наилучшем случае Sp(n)=p и Ep(n)=1. При практическом применении данных показателей для оценки эффективности параллельных вычислений следует учитывать два важных момента:

* При определенных обстоятельствах *ускорение* может оказаться больше числа используемых процессоров Sp(n)>p - в этом случае говорят о существовании *сверхлинейного (superlinear)* ускорения. Несмотря на парадоксальность таких ситуаций ( *ускорение* превышает число процессоров), на практике сверхлинейное *ускорение* может иметь место. Одной из причин такого явления может быть неодинаковость условий выполнения последовательной и параллельной программ. Например, при решении задачи на одном процессоре оказывается недостаточно оперативной памяти для хранения всех обрабатываемых данных и тогда становится необходимым использование более медленной внешней памяти (в случае же использования нескольких процессоров оперативной памяти может оказаться достаточно за счет разделения данных между процессорами). Еще одной причиной сверхлинейного ускорения может быть нелинейный характер зависимости сложности решения задачи от объема обрабатываемых данных. Так, например, известный алгоритм *пузырьковой сортировки* характеризуется квадратичной зависимостью количества необходимых операций от числа упорядочиваемых данных. Как результат, при распределении сортируемого массива между процессорами может быть получено *ускорение*, превышающее число процессоров (более подробно данный пример рассматривается в ["Параллельные методы сортировки"](https://intuit.ru/studies/courses/1156/190/lecture/4958)). Источником сверхлинейного ускорения может быть и различие вычислительных схем последовательного и параллельного методов,
* При внимательном рассмотрении можно обратить внимание, что попытки повышения качества параллельных вычислений по одному из показателей (ускорению или эффективности) могут привести к ухудшению ситуации по другому показателю, ибо показатели качества параллельных вычислений являются часто противоречивыми. Так, например, повышение ускорения обычно может быть обеспечено за счет увеличения числа процессоров, что приводит, как правило, к падению эффективности. И наоборот, повышение эффективности достигается во многих случаях при уменьшении числа процессоров (в предельном случае идеальная *эффективность* Ep(n)=1 легко обеспечивается при использовании одного процессора). Как результат, разработка методов параллельных вычислений часто предполагает выбор некоторого компромиссного варианта с учетом желаемых показателей ускорения и эффективности.

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

Cp=pTp.

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

Далее для иллюстрации введенных понятий в следующем пункте будет рассмотрен учебный пример решения задачи вычисления частных сумм для последовательности числовых значений. Кроме того, данные показатели будут использоваться для характеристики эффективности всех рассматриваемых в лекциях 6 – 11 параллельных алгоритмов при решении ряда типовых задач вычислительной математики.