# Л6. Оценка максимально достижимого параллелизма и масштабируемости параллельных вычислений


1. Гергель В. П. Теория и практика параллельных вычислений. – М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория Базовых Знаний, 2007. – 427 с.
2. Параллельное программирование в стандарте MPI: Учебно-методическое пособие по выполнению лабораторных работ / В.М.Баканов, Д.В.Осипов. - М.: МГУПИ, 2006. –78 с.: ил.

# 1. 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. Так, минимально достижимое время параллельного вычисления частных сумм числовых значений составляет $\log_2 n$. Существует ряд закономерностей, которые могут быть чрезвычайно полезны при построении оценок максимально достижимого параллелизма.

# 2. Закон Амдаля

Пусть $f$ есть доля последовательных вычислений в применяемом алгоритме обработки данных, тогда в соответствии с законом Амдаля (*Amdahl*) ускорение процесса вычислений при использовании p процессоров ограничивается величиной

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

при наличии всего 10% последовательных команд в выполняемых вычислениях, эффект использования параллелизма не может превышать 10-кратного ускорения обработки данных.

# 3. Пример модифицированной каскадной схемы

Для рассмотренного учебного примера вычисления суммы значений для модифицированной каскадной схемы доля последовательных расчетов составляет

$$f = \log_2 n / n$$

и, как результат, величина возможного ускорения ограничена оценкой

$$S^* = n / \log_2 n \, .$$

# 4. Подходы к преодолению ограничения

1. Доля последовательных действий характеризует не возможность параллельного решения задач, а свойства применяемых алгоритмов. Доля последовательных вычислений может быть существенно снижена при выборе более подходящих для распараллеливания методов.
2. Закон Амдаля предполагает, что доля последовательных расчетов $f$ является постоянной величиной и не зависит от параметра $n$, определяющего размерность решаемой задачи. Однако для большого ряда задач доля $f=f(n)$ является убывающей функцией от n, и в этом случае ускорение для фиксированного числа процессоров может быть увеличено за счет увеличения размерности решаемой задачи: ускорение $Sp = Sp\ (n)$ является возрастающей функцией от параметра n (эффект Амдаля).

# 5. Пример вычисления суммы при использовании фиксированного числа процессоров *p*

Суммируемый набор данных делится на блоки размера ***n/p***, для которых сначала параллельно вычисляются частные суммы, а далее эти суммы складываются при помощи каскадной схемы. Длительность последовательной части выполняемых операций (минимально-возможное время параллельного исполнения) в этом случае составляет

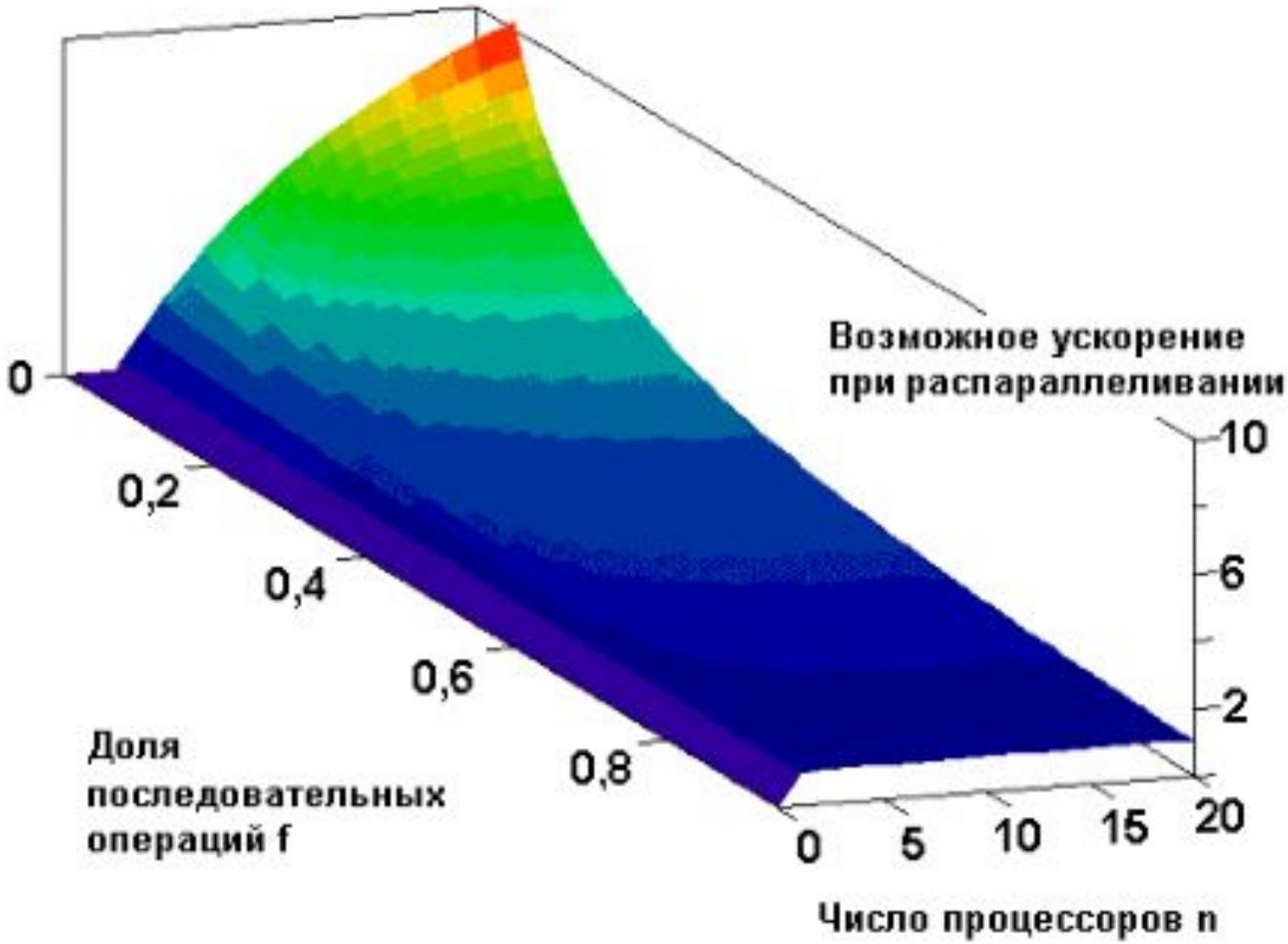
$$T_p = (n/p) + \log_2 p\,,$$

при этом доли последовательных расчетов составляет

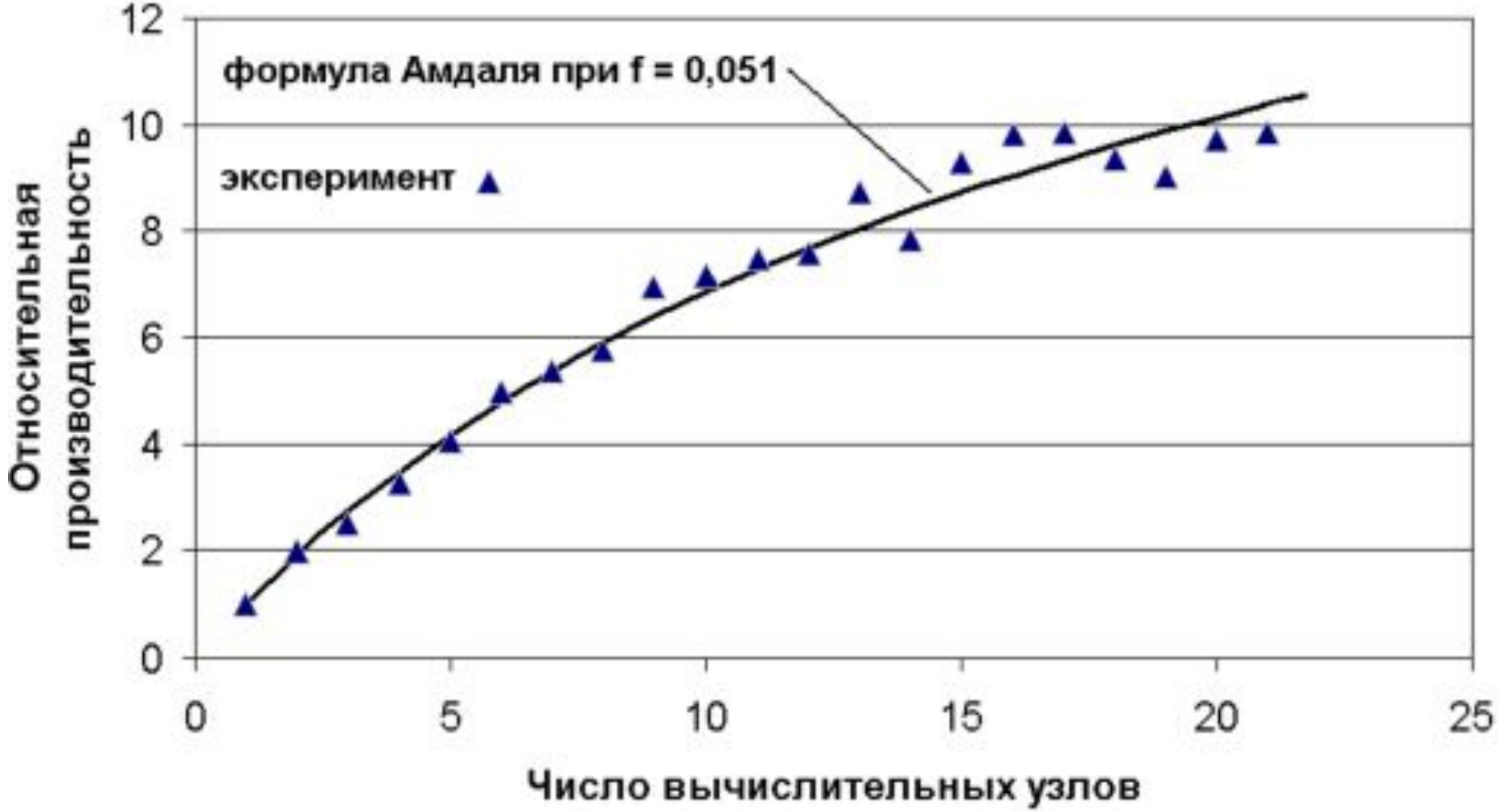
$$f = (1/p) + \log_2 p / n\,.$$

Как следует из полученного выражения, доля последовательных расчетов ***f*** убывает с ростом ***n*** и в предельном случае стремится к идеальной оценке максимального ускорения ***S*=p.***

# 6. Определения *f* по экспериментальным данным



# 7. Эксперимент и расчет по формуле Амдаля f=0,051



# 8. Сетевой закон Амдаля [2]

$$S = \frac{1}{f + \left(\frac{1-f}{n}\right) + c},$$

где $c$ – коэффициент сетевой деградации вычислений,

$c = c_w \times c_t = \frac{W_c \times t_c}{W \times t}$, W c - количество передач данных, W – общее число вычислений в задаче, $t_c$ - время одной передачи данных, $t$ – время выполнения одной операции. Сомножитель

$c_w = \frac{W_c}{W}$ (повышение коего снижает S) определяет составляющую коэффициента деградации, вызванную свойствами алгоритма распараллеливания; $c_t = \frac{t_c}{t}$ - зависящую от соотношения производительности процессора и аппаратуры сети ('техническую') составляющую; значения $c_w$ и $c_t$ могут быть оценены заранее.

# 9. Закон Густавсона - Барсиса.

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

Пусть где τ(n) и π(n) есть времена последовательной и параллельной частей выполняемых вычислений соответственно, т.е.

$$T_1 = \tau(n) + \pi(n),\ T_p = \tau(n) + \pi(n)/p.$$

Используя g можно получить:

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

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

что приводится к виду закона Густавсона-Барсиса:

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

# 10. Применение к суммированию значений

При использовании p процессоров время параллельного выполнения, как уже отмечалось выше, составляет

$T_p = (n/p) + \log_2 p$, что соответствует последовательной доле

$$g = \frac{\log_2 p}{(n/p) + \log_2 p}.$$

За счет увеличения числа суммируемых значений величина ***g*** может быть пренебрежимо малой, обеспечивая получение идеального возможного ускорения $S_p = p$.

# 11. Применение закона Густавсона-Барсиса

При увеличении числа используемых процессоров темп уменьшения времени параллельного решения задач может падать (после превышения определенного порога). Но за счет уменьшения времени вычислений сложность решаемых задач может быть увеличена (например, при суммировании может быть увеличено количество значений). Оценка ускорения при помощи сформулированных закономерностей полезна, т.к. решение более сложных вариантов задач на одном процессоре может оказаться трудоемким и даже невозможным, например, в силу нехватки оперативной памяти. С учетом указанных обстоятельств оценку ускорения, получаемую в соответствии с законом Густавсона-Барсиса, называют ускорением масштабирования (scaled speedup), эта характеристика может показать, насколько эффективны могут быть параллельные вычисления при увеличении сложности решаемых задач.

# 12. Масштабируемость параллельных вычислений

Целью применения параллельных вычислений во многих случаях является не только уменьшение времени выполнения расчетов, но и обеспечение возможности решения более сложных задач, решение которых не возможно при использовании одно процессорных вычислительных систем. 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.

# 13. Анализ масштабируемости параллельных вычислений

Пусть накладные расходы (total overhead) при выполнении параллельного алгоритма составляют

$$T_0 = pT_p - T_1 .$$

Тогда новые выражения для времени параллельного решения задачи, и ускорения и эффективности имеют вид:

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

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

# 14. Сохранение уровня эффективности

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

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

Зависимость ***n=F(p)*** между сложностью решаемой задачи и числом процессоров обычно называют функцией изоэффективности (isoefficiency function)

# 15. Функция изоэффективности суммирования чисел

В этом случае накладные расходы составляют

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

и функция изоэффективности имеет вид $n = Kp\log_2 p$.

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. При увеличении числа процессоров с p до $q$ $(q>p)$ для обеспечения пропорционального роста ускорения $(Sq/Sp)$=(q/p) необходимо увеличить число суммируемых значений n в $(q\log_2 q)/(p\log_2 p)$ раз.