ХАРАКТЕРИСТИКИ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ И СИСТЕМ

Характеристики, связанные с работой вычислительной системы, — производительность, загруженность, ускорение — позволяют оценить качество процессов работы, находить и устранять узкие места.

Пусть вычислительная система состоит из s процессоров[1] с пиковыми производительностями $\pi_1, \ldots, \pi_s$.

Будем рассматривать характеристики в рамках абстрактной модели функционирования вычислительной системы: все операции алгоритма обеспечены необходимыми для срабатывания данными и одинаковы по длительности выполнения. Чтобы не загромождать формулы и выкладки, число выполняемых в единицу времени операций будем считать целым.

Утверждение 1. *Если процессоры работают с загруженностями* $p_1, \ldots, p_s$, *то реальная производительность* r *системы выражается формулой*

$$r = \sum_{i=1}^{s} p_i \pi_i. \tag{1}$$

Доказательство. Реальная производительность системы равна сумме реальных производительностей всех процессоров. Пусть за время T процессор с пиковой производительностью π_i выполнил N_i операций. Тогда время реальной работы процессора равно $\dfrac{N_i}{\pi_i}$, а загруженность процессора равна $p_i = \dfrac{N_i/\pi_i}{T}$. Количество операций, реально выполняемых системой за единицу времени:

$$r = \sum_{i=1}^{s} r_i = \sum_{i=1}^{s} \frac{N_i}{T} = \sum_{i=1}^{s} \frac{N_i/\pi_i}{T} \pi_i = \sum_{i=1}^{s} p_i \pi_i. \qquad \square$$

Из равенства $r = p\pi$ для одного процессора следует, что для повышения производительности надо увеличивать загруженность процессора (что вполне логично). Желательно определение загруженности вычислительной системы ввести таким образом, чтобы равенство $r = p\pi$ также выполнялось.

Определим загруженность системы как взвешенную сумму загруженностей отдельных процессоров:

$$p = \sum_{i=1}^{s} \alpha_i p_i, \quad \alpha_i = \frac{\pi_i}{\sum\limits_{j=1}^{s} \pi_j}.$$

Такое определение согласуется с ранее введенными понятиями: если система состоит из одного процессора ($s = 1$), то понятия загруженности системы и процессора совпадают; загруженность системы, состоящей из одинаковых

[1] Под процессором понимается одно вычислительное ядро.

процессоров $\left(\alpha_i = \frac{1}{s}\right)$ — среднее арифметическое загруженностей всех процессоров; $0 \leq p \leq 1$, причем загруженность системы равна 1 ($p = 1$) тогда и только тогда, когда равны 1 загруженности каждого процессора ($p_i = 1$).

Утверждение 2. *Реальная производительность системы с пиковой производительностью π выражается формулой*

$$r = p\pi. \tag{2}$$

Доказательство. Так как $\pi = \sum_{j=1}^{s} \pi_j$ (пиковая производительность системы равна сумме пиковых производительностей всех процессоров), то
$r = \sum_{i=1}^{s} p_i \pi_i = \pi \sum_{i=1}^{s} p_i \frac{\pi_i}{\sum_{j=1}^{s} \pi_j} = \pi \sum_{i=1}^{s} p_i \alpha_i = \pi p.$ □

Таким образом, для достижения наибольшей реальной производительности вычислительной системы нужно обеспечить наибольшую ее загруженность. Для практических целей понятие производительности является наиболее важным, так как показывает, насколько эффективно система выполняет полезную работу. Понятие загруженности полезно тем, что указывает путь повышения производительности.

Обозначим через π_{max} наибольшую , а через π_{min} наименьшую из величин $\pi_1, \ldots, \pi_s$.

Определим ускорение R реализации алгоритма на данной вычислительной системе:

$$R = \frac{r}{\pi_{max}}. \tag{3}$$

R показывает, во сколько раз количество операций, реально выполняемых системой за единицу времени, превосходит количество операций, которое может быть выполнено за единицу времени одним самым мощным процессором системы. Это определение равносильно определению

$$R = \frac{T_{min}}{T},$$

где T_{min} — время выполнения алгоритма одним полностью загруженным процессором с пиковой производительностью π_{max}, T — время выполнения алгоритма системой[2]. Действительно, если N — число операций алгоритма, то $\frac{T_{min}}{T} = \frac{N/\pi_{max}}{N/r} = \frac{r}{\pi_{max}}$.

[2]На практике используется формула $R = \frac{T_{real}}{T}$, где T_{real} — фактическое время выполнения алгоритма одним процессором.

Из определения ускорения $R = \dfrac{r}{\pi_{max}}$ и формулы (1) $r = \sum\limits_{i=1}^{s} p_i \pi_i$ имеем

$$R = \frac{\sum\limits_{i=1}^{s} p_i \pi_i}{\pi_{max}}. \tag{4}$$

В частности, если все процессоры имеют одинаковые пиковые производительности, то

$$R = \sum_{i=1}^{s} p_i. \tag{5}$$

Формула (4) показывает, что ускорение системы, состоящей из s процессоров, не превосходит s, и может достигать s только в том случае, когда все процессоры имеют одинаковые пиковые производительности и полностью загружены.

С помощью формулы (5) можно установить равносильность понятий загруженности и эффективности вычислительной системы, состоящей из s одинаковых процессоров. Для этого достаточно сравнить определения: загруженность — среднее арифметическое загруженностей всех процессоров, эффективность — отношение ускорения (равного по формуле (5) сумме загруженностей) к s.

Сделаем еще следующее замечание. В формуле $R = \dfrac{T_{min}}{T}$ величина $T_{min} = N/\pi_{max}$ — это время выполнения алгоритма одним полностью загруженным процессором Pr_{max} с теоретической производительностью π_{max}. Это время меньше реального времени выполнения алгоритма одним процессором, так как процессор не может быть полностью загружен хотя бы из-за необходимости обращения к иерархической памяти. В то же время, загруженность процессоров системы может быть больше реальной загруженности процессора Pr_{max} (выполняющего последовательную версию алгоритма), если обменов мало, а работа с иерархической памятью получилась более эффективной. Предпосылками такой работы являются увеличение суммарной распределенной памяти и уменьшение размера массивов, необходимых одному процессору. Поэтому ускорение может быть суперлинейным и быть больше s, если вместо T_{min} при определении ускорения использовать реальное время выполнения алгоритма и определять ускорение как $R = \dfrac{T_{real}}{T}$ — отношение времени выполнения алгоритма на одном процессоре к времени параллельного выполнения.

Утверждение 3. *Если вычислительная система состоит из s связанных между собой процессоров с пиковыми производительностями $\pi_1, \ldots, \pi_s$, то в общем случае*

- *производительность системы не превосходит*

$$r^{max} = s\pi_{min}, \tag{6}$$

- *загруженность системы не превосходит*

$$p^{max} = \frac{s\pi_{min}}{\sum\limits_{j=1}^{s} \pi_j}, \tag{7}$$

- *ускорение системы не превосходит*

$$R^{max} = \frac{s\pi_{min}}{\pi_{max}}. \tag{8}$$

Доказательство. Пусть процессор Pr_i за время T выполняет N_i операций, процессор Pr_j выполняет N_j операций, и каждый результат процессора Pr_i является одним из аргументов очередного срабатывания процессора Pr_j. Тогда количество операций, реализованных процессором Pr_j за время T, не может более, чем на 1 отличаться от количества операций, реализованных процессором Pr_i, т.е. $N_i - 1 \le N_j \le N_i + 1$. Аналогично, для любых процессоров Pr_k и Pr_l, $1 \le k, l \le s$, связанных опосредовано, выполняется $N_l - q \le N_k \le N_l + q$, где q — число соединений между процессорами системы.

Имеем:

$$r = \sum_{i=1}^{s} r_i = \frac{N_1}{T} + \sum_{i=2}^{s} \frac{N_i}{T},$$

$$\frac{N_1}{T} + \sum_{i=2}^{s} \frac{N_1 - q}{T} \le r \le \frac{N_1}{T} + \sum_{i=2}^{s} \frac{N_1 + q}{T},$$

$$\frac{N_1 s}{T} - \frac{q(s-1)}{T} \le r \le \frac{N_1 s}{T} + \frac{q(s-1)}{T}.$$

Так как $\dfrac{q(s-1)}{T}$ стремится к нулю при T, стремящимся к бесконечности, то асимптотически $r = \dfrac{N_1 s}{T}$. Асимптотически все N_k равны между собой; кроме того, всегда выполняется неравенство $N_k \le \pi_k T$. Поэтому асимптотически $N_1 \le \pi_{min} T$, а максимально возможная реальная производительность равна $\pi_{min} s$. Справедливость формулы (6) доказана.

Справедливость формулы (7) $p^{max} = \dfrac{s\pi_{min}}{\sum\limits_{j=1}^{s} \pi_j}$ непосредственно следует из формул (2) $r = p\pi$, (6) $r^{max} = s\pi_{min}$, а справедливость формулы (8) $R^{max} = \dfrac{s\pi_{min}}{\pi_{max}}$ непосредственно следует из формул (3) $R = \dfrac{r}{\pi_{max}}$, (6) $r^{max} = s\pi_{min}$. □

Из доказательства утверждения следует, что для системы связанных процессоров

- асимптотически каждый процессор выполняет одно и то же число операций.

Поэтому:

- загруженность любого процессора не превосходит загруженности самого непроизводительного процессора,
- если загруженность какого-то процессора равна 1, то это самый непроизводительный процессор.

Еще одно следствие утверждения 3 является именным:

Первый закон Амдала. *Производительность многопроцессорной вычислительной системы определяется в общем случае самым непроизводительным процессором.*

Далее исследуем, как выражается ускорение системы, если часть операций алгоритма выполняется последовательно.

Предположим, что n из N операций алгоритма выполняются системой последовательно. Отношение $\beta = n/N$ называется долей последовательных вычислений.

Второй закон Амдала. *Пусть система состоит из s одинаковых процессоров. Если при вычислении параллельной части алгоритма все процессоры загружены полностью, то ускорение выражается формулой*

$$R = \frac{s}{\beta s + (1-\beta)}. \tag{9}$$

Доказательство. Воспользуемся формулой (5) $R = \sum\limits_{i=1}^{s} p_i$. Пусть всего выполняется N операций. Не ограничивая общности можно считать, что последовательные βN операций выполняются на первом процессоре, и $N - \beta N$ операций выполняется параллельно на s процессорах по $(1-\beta)N/s$ операций на каждом. Ясно, что $p_1 = 1$. Найдем остальные p_i.

Всего алгоритм реализуется за время $T_1 = \dfrac{\beta N + (1-\beta)N/s}{\pi_1}$. Параллельная часть алгоритма реализуется за время $T_i = \dfrac{(1-\beta)N/s}{\pi_i}$. В выражениях T_1 и T_i учтено, что при вычислении параллельной части алгоритма все процессоры загружены полностью. Так как все процессоры имеют одинаковые пиковые производительности, то

$$p_i = \frac{T_i}{T_1} = \frac{(1-\beta)N/s}{\beta N + (1-\beta)N/s} = \frac{1-\beta}{\beta s + (1-\beta)}.$$

Следовательно,

$$R = 1 + \sum_{i=2}^{s} p_i = 1 + \sum_{i=2}^{s} \frac{1-\beta}{\beta s + (1-\beta)} = 1 + \frac{(1-\beta)(s-1)}{\beta s + (1-\beta)} = \frac{s}{\beta s + (1-\beta)}. \quad \square$$

Прямым следствием второго закона Амдала является следующее очень важное утверждение.

Третий закон Амдала. *Пусть вычислительная система состоит из одинаковых процессоров. При любом режиме работы ее ускорение не может превзойти обратной величины доли последовательных вычислений.*

Например, если доля последовательных вычислений составляет всего 1%, то ни при каком числе процессоров нельзя добиться ускорения более чем в 100 раз. Если вычислительная система состоит из тысяч процессоров, то для эффективного ее использования доля последовательных вычислений должна быть порядка десятых и сотых долей процента.

Теперь исследуем, как изменится формула (9), если учитывать затраты времени на коммуникационные операции (между вычислительными процессами). Введем в рассмотрение величины $c_{alg} = N_{comm}/N$ и $c_{dev} = t_{comm}/\tau$, где N_{comm} – число коммуникационных операций алгоритма, t_{comm} – время выполнения одной коммуникационной операции, τ – время выполнения одной вычислительной операции. Величина c_{alg} определяется алгоритмом, а величина c_{dev} – техническими характеристиками вычислительной системы и способом реализации коммуникационных операций.

Сетевой закон Амдала. *Пусть система состоит из s одинаковых процессоров. Если при вычислении параллельной части алгоритма все процессоры загружены полностью, то ускорение выражается формулой*

$$R = \frac{s}{\beta s + (1-\beta) + c_{alg}c_{dev}s}. \tag{10}$$

Доказательство. Пусть T – время реализации алгоритма на одном процессоре (тогда $T = N\tau$). Из формулы (9) $R = \dfrac{s}{\beta s + (1-\beta)}$, записанной в виде $R = \dfrac{T}{(\beta + (1-\beta)/s)T}$, следует, что $(\beta + (1-\beta)/s)T$ есть время параллельной реализации алгоритма без учета затрат на коммуникационные операции. Тогда время параллельной реализации алгоритма с учетом затрат на коммуникационные операции равно $(\beta + (1-\beta)/s)T + N_{comm}t_{comm}$. В этом случае

$$R = \frac{T}{(\beta + (1-\beta)/s)T + N_{comm}t_{comm}} = \frac{1}{(\beta + (1-\beta)/s) + (N_{comm}t_{comm})/T} =$$

$$\frac{s}{\beta s + (1-\beta) + s(N_{comm}t_{comm})/(N\tau)} = \frac{s}{\beta s + (1-\beta) + c_{alg}c_{dev}s}.$$

□

Из формулы (10) следует

Сетевой аналог третьего закона Амдала. *Ускорение не может превзойти величины*

$$\frac{1}{\beta + c_{alg}c_{dev}},$$

где β *– доля последовательных вычислений алгоритма,* c_{alg} *– отношение числа всех коммуникационных операций алгоритма к числу всех вычислительных операций алгоритма,* c_{dev} *– отношение времени выполнения одной коммуникационной операции к времени выполнения одной вычислительной операции.*

Основной ресурс увеличения ускорения при организации распределенных вычислений алгоритма – уменьшение числа коммуникационных операций.

Получим еще одно представление ускорения. Обозначим через β_T долю времени последовательных вычислений при условии, что параллельная часть полностью занимает s одинаковых процессоров. Не следует путать долю времени последовательных вычислений β_T (это величина по определению зависит от числа s используемых процессоров) и долю последовательных вычислений β (это величина по определению не зависит от числа используемых процессоров).

Закон Густавсона-Барсиса. *Пусть система состоит из* s *одинаковых процессоров. Если при вычислении параллельной части алгоритма все процессоры загружены полностью, то ускорение выражается формулой*

$$R = s - (s-1)\beta_T. \tag{11}$$

Доказательство. Пусть, как и ранее, всего выполняется N операций, n операций выполняются в последовательной части, $\beta = n/N$ – доля последовательных вычислений, последовательные βN операций выполняются на первом процессоре, $N - \beta N$ операций выполняется параллельно на s процессорах.

Всего алгоритм реализуется за время $T_1 = \dfrac{\beta N + (1-\beta)N/s}{\pi_1}$. Последовательная часть алгоритма реализуется за время $\dfrac{\beta N}{\pi_1}$. Доля времени последовательных вычислений: $\beta_T = \dfrac{\beta N}{\beta N + (1-\beta)N/s} = \dfrac{\beta s}{\beta s + (1-\beta)}$.

Отсюда вытекает соотношение, выражающее связь между β и β_T:

$$\beta = \frac{\beta_T}{s - (s-1)\beta_T}.$$

Подставив полученное представление β в формулу Амдала (9) $R = \dfrac{s}{\beta s + (1-\beta)}$, получим формулу Густавсона-Барсиса (11). □

Отметим принципиальное отличие случаев применения формул (9)

$$R = \frac{s}{\beta s + (1 - \beta)}.$$

и (11)

$$R = s - (s - 1)\beta_T.$$

Формула Амдала применяется для прогноза возможного ускорения. При этом величину β можно подсчитать, не пропуская программу на параллельном компьютере (или вообще не пропуская программу ни на каком компьютере). Формула Густавсона-Барсиса применяется для оценивания достигнутого ускорения, не пропуская программу на одном процессоре. При этом величину β_T можно подсчитать в процессе решения задачи на параллельном компьютере.

Список использованных источников

1. Воеводин В. В., Воеводин Вл. В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2002. – 608 с.

2. Воеводин В. В. Вычислительная математика и структура алгоритмов. – Москва: МГУ, 2006. – 112 с. http://parallel.ru/info/parallel/voevodin/

3. Шпаковский Г.И., Верхотуров А.Е., Серикова Н.В. Руководство по работе на вычислительном кластере. – Минск: БГУ, 2004. – 171 с.