ОЦЕНКИ ВРЕМЕНИ В МОДЕЛИ ПАРАЛЛЕЛЬНОГО ВЫПОЛНЕНИЯ ПРОГРАММ

В.А. Биллиг

Тверской государственный технический университет

Рассматривается модель параллельных вычислений в вычислительной системе с общей памятью. Выполняемая программа рассматривается как множество модулей, связанных по данным. Граф зависимостей отражает эту связь. Целью работы является получение оценок времени выполнения программы одним процессором – T_1, конечным числом процессоров –T_p и для идеализированного случая, когда число процессоров не ограничивается – T_∞. Оценки получены для случая, когда времена выполнения модулей программы различны.

Параллельные вычисления становятся одним из магистральных направлений развития информационных технологий. Можно указать на две причины, определяющие важность этого направления. Первая состоит в том, что стратегически важные для развития государства задачи могут быть решены только с применением суперкомпьютеров, обладающих сотнями тысяч процессоров, которые нужно заставить работать одновременно. Вторая причина связана с другим полюсом компьютерной техники, на котором находятся обычные компьютеры, ориентированные на массового пользователя. И эта техника становится многоядерной, и ее требуется эффективно использовать, так что параллельные вычисления требуются и здесь.

Для поддержки параллельных вычислений сделано достаточно много, от архитектуры вычислительных систем, операционных систем, языков программирования до разработки специальных параллельных алгоритмов. Тем не менее для программиста, решающего сложную задачу, построение и отладка эффективной параллельной программы все еще остается не простым занятием. В ключевом докладе В.В. Воеводина [1] отмечалось, что к началу активного внедрения вычислительных систем параллельной архитектуры в практику решения больших прикладных задач нужный теоретический фундамент не был построен, так же как и не был развит математический аппарат исследований.

Целью данной работы является рассмотрение одной из моделей параллельного вычисления. Для этой модели мы хотим получить оценки времени выполнения программы одним процессором – T_1, конечным числом процессоров –T_p и для идеализированного случая, когда число процессоров не ограничивается – T_∞.

Рассмотрим программу P, состоящую из n модулей:

$$P = \{M_1, M_2, \ldots M_n\}$$

Будем предполагать, что программа P выполняется на компьютере, обладающем некоторым числом процессоров, работающих на общей памяти. Выходные данные, полученные в результате работы модуля M_i, могут являться входными данными для модуля M_j. Так естественным образом возникает зависимость между модулями, определяющая возможный порядок их выполнения.

Множество модулей разобьём на k уровней. К уровню i отнесем те модули, для начала работы которых требуется завершение работы модулей, из которых хотя бы один принадлежит уровню $i - 1$. Модуль уровня i с номером k будем обозначать как M_k^i.

Модули, принадлежащие уровню 1, имеют все необходимые данные, полученные от внешних источников. Они не требуют завершения работы других модулей и в принципе могут выполняться параллельно, будучи запущенными в начальный момент выполнения программы.

Свяжем с программой P ориентированный граф зависимостей модулей. Граф не содержит циклов и отражает разбиение модулей на уровни. Модули являются вершинами графа, а дуги отражают зависимости между модулями. Дуга ведет от модуля M_k^i к модулю M_l^j, если для начала выполнения модуля M_k^i требуется завершение работы модуля M_l^j. В узлах графа содержится информация об ожидаемом времени выполнения модуля, где время измеряется в некоторых условных единицах. На рис.1 показан пример графа зависимостей.

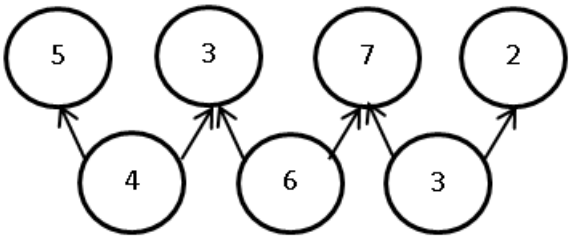

Рис. 1. Пример графа зависимостей

Обозначим через T_1 время, требуемое для выполнения программы P одним процессором, T_p – p процессорами, T_∞ – время, требуемое в случае, когда число процессоров неограниченно. В последнем случае достаточно n процессоров, по числу модулей нашей программы.

Предполагается, что все эти характеристики рассчитываются при соблюдении двух условий:

- Выполняются зависимости между модулями, заданные графом зависимостей.
- Характеристики вычислены для оптимального расписания работы процессоров.

В случае одного процессора достаточно выполнения только первого условия. Обычно предполагается естественный порядок выполнения модулей – последовательное выполнение модулей одного уровня, затем переход к выполнению модулей следующего уровня.

Для случая неограниченного числа процессоров оптимальным является такое расписание, когда каждый модуль начинает выполняться, как только завершены все модули, необходимые для его работы.

Для случая p процессоров можно распределить модули по процессорам, задав для каждого процессора P_i множество модулей, выполняемых этим процессором:

$$D(P_i) = \{M_{i,1}, M_{i,2}, \dots M_{i,r}\}.$$

Распределение модулей по процессорам совместно с графом зависимостей однозначно определяет расписание работ и время выполнения программы при данном расписании. Предполагается, что каждый процессор выполняет модули из распределения $D(P_i)$. После завершения очередного модуля он сразу же переходит к выполнению следующего модуля, если для этого модуля выполнены все зависимости, заданные графом зависимостей. В противном случае процессор ждет окончания работы требуемых модулей. Время завершения последнего модуля в распределении $D(P_i)$ задает время работы данного процессора. Тот процессор, который последним заканчивает работу, и определяет общее время решения задачи T_p^D для данного расписания. Введенная ранее характеристика T_p предполагает оптимальное расписание:

$$T_p = \min_D T_p^D.$$

Задача составления оптимального расписания относится к сложным задачам [3]. На практике для программ большого размера не удается явно вычислить значение T_p. По этой причине несомненный интерес представляет получение оценок для T_p.

Для введенных характеристик выполняется естественное соотношение:

$$T_\infty \le T_p \le T_1. \quad (1)$$

Нас будет интересовать получение более точных оценок для T_p. В работе В.П. Гергеля [2] показано, что для случая, когда время выполнения всех модулей одинаково, имеют место следующие соотношения:

$$\frac{T_1}{p} \le T_p \le \frac{T_1}{p} + T_\infty. \quad (2)$$

Рассмотрим более интересный для практики случай, когда модули программы для своего выполнения требуют разного времени. Пусть M^i – множество модулей уровня i:

$$M^i = \{M_1^i, M_2^i, \dots M_{k_i}^i\}. \quad (3)$$

Для каждого из этих модулей известно время, требуемое на его выполнение, – $t_{i,j}$.

И в этом случае нетрудно рассчитать время T_1 – время, требуемое на выполнение всей работы одним процессором:

$$T_1 = \sum_{i=1}^{k} \sum_{j=1}^{k_i} t_{i,j}. \quad (4)$$

Как рассчитать время T_∞ в этой ситуации, когда мы располагаем неограниченным числом процессоров? Введем для каждого модуля время окончания его работы – t_j^i. Это время будем рассчитывать по следующей формуле:

$$t_j^i = t_{i,j} + t_{jmax}^{i-1}. \quad (5)$$

Здесь t_{jmax}^{i-1} – время окончания работы того модуля уровня i-1, который:

- необходим для работы модуля M_j^i ;
- из всех необходимых модулей завершает свою работу последним.

Тогда время T_∞ можно рассчитать следующим образом:

$$T_\infty = \max_j\{t_j^k\} \quad (6)$$

Формула (6) говорит, что время завершения последнего модуля уровня k и является временем T_∞ при оптимальном расписании работ. Справедлива следующая теорема:

Теорема 1

Время T_∞ задается максимально нагруженным путем в графе зависимостей.

Оценка снизу

Лемма 1

Для T_p справедлива оценка:

$$T_p \ge \frac{T_1}{p}. \quad (7)$$

Оценка сверху

Пусть работу выполняют p процессоров. Составление расписания означает, что граф зависимостей разбивается на p непересекающихся подграфов. Все модули каждого из подграфов выполняются одним процессором. Подграф с максимальным временем выполнения для данного разбиения будем называть максимально нагруженным подграфом. Оптимальное расписание предполагает такое разбиение, при котором максимально нагруженный подграф выполняется за минимально возможное время.

Лемма 2

Для T_p справедлива оценка:

$$T_p \le \frac{T1}{p} + T_\infty. \quad (8)$$

Дадим графическую интерпретацию. Задание нижней и верхней оценки для $T_p(p)$ означает, что эта функция ограничена двумя гиперболами. Функция убывающая. В начальной точке при p =1 по определению $T_p(1) = T_1$, так что функция находится в за-

данном коридоре. Это же справедливо и для конечных точек, для всех p, больших некоторого значения p^*, при котором $T_p = T_\infty$. Рисунок 2 иллюстрирует поведение T_p.

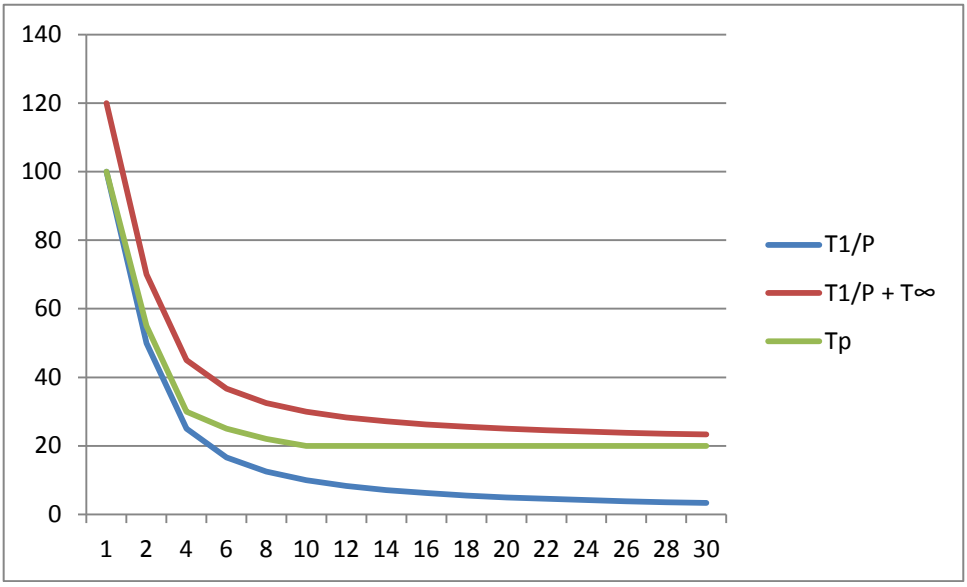

Рис. 2. Поведение функции $T_p(p)$

В заключение дадим некоторые практические рекомендации, следующие из полученных оценок. Выигрыш, который можно получить, используя дополнительные процессоры, зависит от разницы между общим временем выполнения всех модулей программы – T_1 и временем выполнения критического пути в графе зависимостей – T_∞.

Эта разница максимальна для крайнего случая, когда все модули могут выполняться независимо и в графе зависимостей все модули находятся на одном первом уровне. Критический путь в этом случае состоит из одного модуля, требующего максимального времени своего выполнения. Так что T_1 – это время выполнения всех модулей, а T_∞ – это время выполнения одного модуля. Привлечение p процессоров может дать существенный эффект, уменьшая время выполнения практически до среднего времени выполнения одного модуля T_1/p.

Эта разница минимальна для другого крайнего случая – строго последовательной программы, когда N модулей программы расположены на N уровнях и критический путь задает выполнение всех модулей. В этом случае T_1 и T_∞ совпадают и, как следствие, T_p равно T_1 при любом числе процессоров, так что привлекать дополнительные процессоры в этом случае бессмысленно.

Для строго последовательной программы дополнительные процессоры не позволяют уменьшить время выполнения программы в сравнении со временем выполнения этой же программы одним процессором. Так, например, задача о «Ханойской башне» на суперкомпьютере с сотнями тысяч процессоров будет решаться столь же долго, как и на компьютере с одним процессором. Это вытекает из сути задачи. Перенос следующего кольца требует завершения переноса предыдущего кольца, параллельно эту работу выполнять нельзя.

Литература

1. Воеводин В.В. Математические проблемы параллельных вычислений // Доклад на Всероссийской конференции «Научный сервис в сети Интернет: технологии распределенных вычислений», 2012.
2. Гергель В.П. Теория и практика параллельных вычислений. М.: Изд. «Бином», 2007.
3. Кормен Т. и др. Алгоритмы. Построение и анализ. Изд. «Вильямс», 2012.