Особенности исполнения фрагментированных программ

В.Э. Малышкин
Институт вычислительной математики и математической геофизики СО РАН, Новосибирск

Рассматриваются проблемы организации эффективного выполнения фрагментированных программ на мультипроцессорах.

В ИВМиМГ СО РАН разрабатывается система фрагментированного программирования для создания библиотеки численных алгоритмов. Основные требования к системе заключаются в следующем:

1. Фрагментированная программа собирается из готовых к исполнению фрагментов вычислений.
2. Фрагментированная структура программы сохраняется в ходе вычислений, т.е. каждый фрагмент вычислений образует независимо исполняющийся процесс.
3. Переменные – это переменные программы, вычисление – именуется каждое срабатывание операции.
4. Все технические детали параллельного программирования должны быть сконцентрированы в системном программном обеспечении .

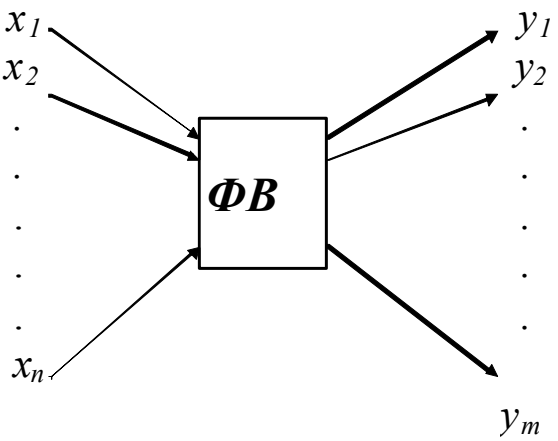


Рис. 1. Обращение к фрагменту вычисления

К фрагменту вычислений можно обратиться, например, как к обычной процедуре в последовательном языке программирования (рис. 1).

Технология, язык и система фрагментированного программирования разрабатываются для того, чтобы автоматически обеспечивать динамические свойства прикладной параллельной программы. Рассматриваются следующие необходимые динамические свойства параллельных программ:

1. При исполнении параллельной программы на мультикомпьютере в каждый его процессорный элемент загружается, по крайней мере, одна программа, а лучше несколько, со своими наборами входных данных для обработки. Таким образом, при исполнении параллельная программа представляется как множество параллельно протекающих и взаимодействующих последовательных процессов.
2. Недетерминизм исполнения. Для последовательного исполнения на множестве всех процессов должен быть установлен линейный порядок. Для параллельного исполнения на множестве процессов прикладной программы устанавливается частичный порядок и процессы исполняются, не нарушая этот порядок. При каждом конкретном исполнении ПП (с одними и теми же входными данными) ее процессы исполняются, вообще говоря, в разном порядке относительно друг друга и в случае ошибки ПП может вырабатывать разные результаты при разных ее исполнениях на одних и тех же данных.
3. Гибкая настройка программы на все доступные ресурсы мультикомпьютера. Серьезная проблема – динамическое распределение работы таким образом, чтобы обеспечить равномерную загрузку ресурсов мультикомпьютера (отображение прикладного алгоритма на ресурсы мультикомпьютера).
4. Динамическая балансировка загрузки. В зависимости от входных данных процессы могут потреблять разное количество ресурсов (например, исполняться за разное время в зависимости от числа итераций внутреннего цикла). В ходе исполнения ПП количество потребляемых процессами ПП ресурсов также может существенно изменяться. Во всех этих случаях нужна динамическая (в ходе исполнения) балансировка загрузки мультикомпьютера.
5. Переносимость в классе мультикомпьютеров. Параллельная программа должна быть хорошо переносима в классе мультикомпьютеров, т.е. хорошо исполняться на каждом мультикомпьютере из некоторого класса. ПП должна хорошо (если можно, оптимально) вы-

полняться на всех тех ресурсах, что ей выделены при текущем исполнении.

6. Динамизм поведения. При реализации численных моделей большого размера ПП должна уметь учитывать поведение модели и подстраиваться под него для достижения высокой производительности.
7. Реализация межпроцессорных коммуникаций на фоне счета.

Фрагментированная программа представляется четырьмя множествами: множество фрагментов данных, множество фрагментов кода, множество фрагментов вычислений и бинарное отношение частичного порядка на множестве фрагментов вычислений. Средства фрагментированного программирования поясним на простых примерах.

Алгоритм умножения матриц

1. Исходный алгоритм:

$$C = A \times B, \quad A = (a_{ij})_{i,j=\overline{1,N}}, \quad B = (b_{ij})_{i,j=\overline{1,N}}, \quad C = (c_{ij})_{i,j=\overline{1,N}},$$

$$c_{i,j} = \sum_{k=1}^{N} a_{i,k} \times b_{k,j}, \quad i, j = 1,2,...,N.$$

2. Фрагментированный алгоритм

$$C_{I,J} = \sum_{L=1}^{K} A_{I,L} \times B_{L,J} = \sum_{L=1}^{L=K} C^{L}{}_{I,J}.$$

Здесь $C_{I,J}, A_{I,L}, B_{L,J}, C^{L}{}_{I,J} \quad I, J, L = 1,2,...,K.$

2.1.Множество фрагментов данных

$C_{I,J}, A_{I,L}, B_{L,J}, C^{L}{}_{I,J} \quad I, J, L = 1,2,...,K.$

2.2.Множество фрагментов вычислений

1) $\Phi_{IJ}^{L} = C_{I,J}^{L} = A_{I,L} \times B_{L,J}$,

2) $\Phi C_{IJ} = C_{I,J} = \sum_{L=1}^{K} C_{I,J}^{L}$.

2.3.Порядок:

$\forall L$ *и фиксированных I и J,*

$\Phi_{IJ}^{L} < \Phi C_{IJ}$.

Для простоты понимания можно представлять себе каждый фрагмент вычислений как процедуру. Для исполнения фрагментированных программ разработана run-time система, которая работает по общему правилу: если все входные переменные фрагмента вычисления получили значения (готовы к исполнению), то фрагмент может быть (но не обязательно будет) назначен на исполнение. В силу задания управления бинарным отношением частичного порядка многие фрагменты будут готовы к исполнению, у run-time системы выбор очередного

фрагмента на исполнение будет слишком велик и нередко может быть сделан неудачно, т.е. качество исполнения ФП будет неприемлемым.

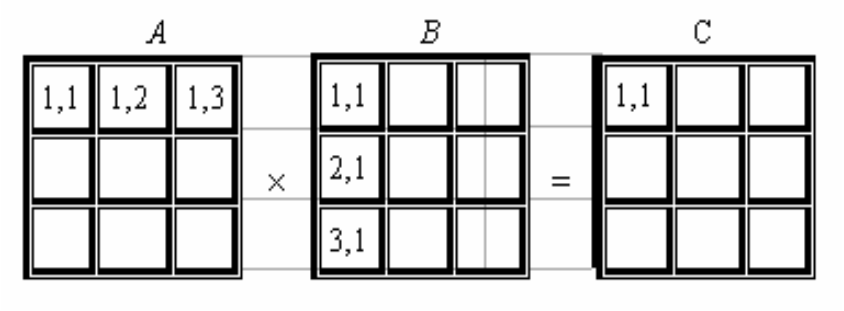


Рис. 2. Фрагментированная программа умножения матриц

Для фрагментированной программы умножения матриц примера такое выполнение фрагментов вычислений, в котором сначала выполняются все C^{L1}_{IJ}, где $L1$=1,2,3, неприемлемо, так как происходит накапливание фрагментов данных C^{L1}_{IJ} для всех I и J без их потребления.

Уже на этом простом примере видны проблемы организации исполнения фрагментированных программ, выбора желаемого фрагмента на исполнение. Аналогичные проблемы возникают и в процессе динамического распределения вычислительных ресурсов мультикомпьютера.

Литература

1. **Valkovskii V.A., Malyshkin V.E.** Parallel Program and System Synthesis on the Basis of Computational Models. Novosibirsk: Nauka, 1988. 128 p.
2. **Kraeva M.A., Malyshkin V.E.** Assembly Technology for Parallel Realization of Numerical Models on MIMD-Multicomputers // Int. Journal on Future Generation Computer Systems. 2001. Vol. 17, No. 6, P. 755-765.
3. **Malyshkin V.E.** Assembling of Modeling Parallel Programs // Handbook of Research on Scalable Computing Technologies. 2009. Chapter 13. IGI Global. P. 295-311.
4. **Malyshkin V.E., Sorokin S.B., Chajuk K.G.** Fragmentation of Numerical algorithms for the Parallel Subroutines Library // In Proceedings of the PaCT-2009 Int. conference, LNCS, 2009.