Методика фрагментации численных алгоритмов для библиотеки параллельных численных подпрограмм*

С.Е. Киреев
Институт вычислительной математики и математической геофизики
СО РАН, Новосибирск
Новосибирский государственный университет

Представлены первые результаты разработки методики фрагментации численных алгоритмов. Выделены принципы построения фрагментированных алгоритмов. Рассмотрен вопрос фрагментации регулярных структур, образуемых объектами алгоритма.

Введение

Часто метод моделирования требует, чтобы реализующая его параллельная программа обладала многими динамическими свойствами, такими как настройка на доступные ресурсы, балансировка загрузки с учетом поведения модели, выполнение вычислений на фоне межузловых обменов данными. Эти задачи относятся к области системного программирования и требуют наличия специальных знаний и навыков у специалистов по численному моделированию.

Технология фрагментированного программирования призвана автоматизировать решение задач системного параллельного программирования при реализации численных моделей. Она основана на методе синтеза параллельных программ [1, 2]. Суть ее состоит в следующем:

- Параллельная программа представляется в виде двух явным образом разделенных слоев:
  - теоретико-множественное описание алгоритма – переносимое, независимое от архитектуры;
  - реализационная часть – исполнительная система, обеспечивающая эффективное исполнение алгоритма на конкретной вычислительной системе.
- Алгоритм представляется в специальном фрагментированном виде: в виде множества фрагментов данных, множества фрагментов вычислений и отношений на них. Разделение на фрагменты сохраняется до самого момента исполнения – это позволяет исполнитель-

* Проект поддержан грантом РФФИ № 10-07-00454а.

ной системе динамически выполнять настройку на имеющиеся ресурсы.

Фрагментированный алгоритм содержит только минимальное управление, определяемое зависимостями по данным, и не содержит распределения ресурсов. Таким образом, он допускает множество способов исполнения, что обеспечивает его переносимость. Задача исполнительной системы – выполнить отображение объектов алгоритма (переменных, операций) на ресурсы конкретной вычислительной системы, автоматически обеспечивая все необходимые динамические свойства параллельной программы [3]. Фрагментация – это технологический прием, позволяющий уменьшить число объектов алгоритма и тем самым упростить задачу построения эффективного распределения ресурсов и управления.

В качестве первого приложения технологии фрагментированного программирования разрабатывается библиотека фрагментированных численных подпрограмм [4] на базе экспериментальной системы фрагментированного программирования LuNA [3]. В настоящий момент выполнена фрагментация ряда алгоритмов матричных операций для плотных и разреженных матриц, алгоритмов вычислений по явной схеме на структурированных сетках, задачи статистического моделирования. Ведется работа над фрагментацией метода частиц-в-ячейках.

Настоящая работа посвящена проблеме построения таких фрагментированных алгоритмов, которые могли бы эффективно исполниться на современных параллельных вычислительных системах при наличии соответствующей исполнительной системы. На основе накопленного опыта фрагментации численных алгоритмов формируется методика построения фрагментированных алгоритмов, которая может быть поддержана исполнительной системой.

Основные определения

Будем пользоваться следующим определением алгоритма. Алгоритм – это совокупность следующих множеств и отношений:

- множество переменных единственного присваивания;
- множество операций единственного срабатывания;
- отношение вход-выход на множествах переменных и операций.

Алгоритм эквивалентным образом можно представить как множество функциональных термов (возможно, потенциально бесконечное).

Пусть задан алгоритм с атомарными (неделимыми) переменными и операциями. Будем называть его исходным алгоритмом. Далее будем

рассматривать процесс построения фрагментированного алгоритма из исходного. В процессе фрагментации происходят следующие преобразования:

- некоторые атомарные переменные исходного алгоритма объединяются в подмножества, каждое из которых формирует одну новую агрегированную переменную;
- некоторые части исходного алгоритма (подмножества переменных и операций) объединяются в новые агрегированные операции, причем переменная может войти в агрегированную операцию, только если все связанные с ней операции (через отношение вход-выход) входят в ту же агрегированную операцию;
- связи вход-выход соответствующим образом объединяются.

Фрагментированный алгоритм – это алгоритм с агрегированными переменными и операциями. Агрегированные переменные будем называть фрагментами данных, агрегированные операции – фрагментами вычислений.

Принципы фрагментации

Исходя из требования эффективного исполнения фрагментированной программы, можно сформулировать следующие принципы построения фрагментированных алгоритмов:

1. Число объектов и связей алгоритма при фрагментации должно уменьшаться. Чем меньше число объектов и связей в алгоритме, тем проще исполнительной системе решить задачу отображения их на ресурсы.
2. Фрагменты алгоритма должны быть как можно более единообразными (во всех смыслах). Единообразие фрагментов алгоритма позволяет использовать одни и те же способы обработки для всех фрагментов, что упрощает устройство исполнительной системы и повышает ее эффективность.
3. Фрагментированный алгоритм должен иметь возможность настраиваться на вычислительную систему. Фрагментацию алгоритма можно выполнять по-разному. Разные способы фрагментации могут быть оптимальными для разных вычислительных систем. Необходимо, чтобы фрагментированный алгоритм имел явно задаваемые параметры, оптимальные значения которых можно выбрать вручную или автоматически для конкретной вычислительной системы. Таким параметром, например, является размер фрагментов. При изменении размера задачи оп-

тимальные значения параметров алгоритма меняться не должны, должно меняться только число фрагментов.

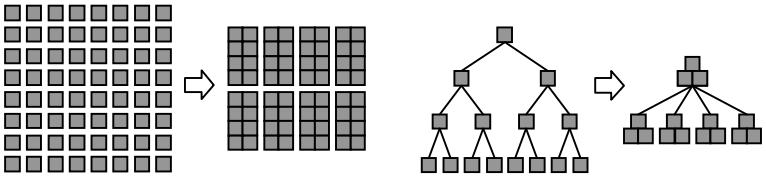
4. Фрагментированный алгоритм должен быть масштабируемым – число фрагментов должно быть пропорционально объему вычислений. Это значит, что если с увеличением размера задачи объем вычислений увеличится в 2 раза, то и число фрагментов должно увеличиться примерно в 2 раза.

Конечно, не каждый алгоритм можно фрагментировать так, чтобы он удовлетворял всем принципам. Но чем больше принципов соблюдено при построении фрагментированного алгоритма, тем лучше будут свойства полученной на его основе фрагментированной программы.

Фрагментация регулярных структур

Большинство численных алгоритмов имеют регулярным образом организованные данные и вычисления. Фрагментацию регулярных структур, образуемых объектами алгоритма, следует рассмотреть отдельно.

Исходя из принципа единообразия, объекты алгоритма должны быть объединены во фрагменты одинаковым образом (рис. 1).



(а) (б)

Рис. 1. Примеры фрагментации регулярных структур алгоритма

В исходном алгоритме подтермы составляют некоторые структуры, каждая из которых имеет, как правило, один тип регулярности. После фрагментации такой структуры типов регулярности становится два. Будем называть их внутренней структурой и внешней структурой:

- Внутренняя структура фрагмента данных – структура данных, образуемая атомарными переменными внутри фрагмента данных.
- Внешняя структура фрагментов – структура, образуемая фрагментами данных или вычислений одной регулярной структуры алгоритма.

Например, на рис. 1, *а* и внешняя структура – двумерный массив 2×4, а внутренняя структура – двумерный массив 4×2. На рис. 1, *б*

внешняя структура – дерево из 5 элементов, а внутренняя – дерево из 3 элементов.

При фрагментации алгоритмов встает задача выбора внешней и внутренней структуры для каждой регулярной структуры исходного алгоритма. Внешняя структура выбирается, прежде всего, исходя из рассмотренных выше принципов. Выбор внутренней структуры относится уже к интерпретации объектов алгоритма. Можно назвать следующие критерии выбора внутренней структуры: минимизация размера фрагмента, минимизация числа операций, наличие оптимизированных библиотечных операций.

Для примера рассмотрим фрагментацию алгоритма умножения разреженной матрицы на вектор. Пусть A – матрица размера $N \times N$, x,y – векторы размера N. Алгоритм умножения произвольной матрицы на вектор описывается формулой $y_i = \sum_{j=1}^{N} a_{ij} x_j$, где a_{ij} – элементы матрицы A; x_j, y_i – элементы векторов x и y.

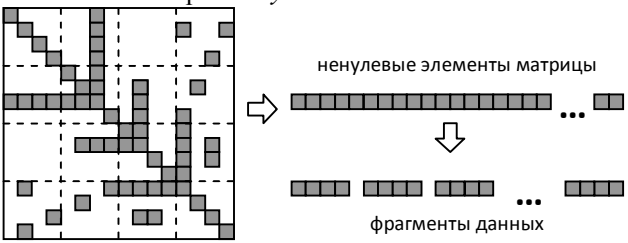


Рис. 2. Фрагментация разреженной матрицы. Способ 1

Пусть вектор x разделен на P фрагментов равного размера. Рассмотрим два способа фрагментации матрицы A. На рис. 2 представлен первый способ фрагментации данных: разделение списка ненулевых элементов матрицы на фрагменты одинакового размера.

Пусть число фрагментов матрицы будет Q. Для записи алгоритма умножения матрицы на вектор потребуется описать $P \times Q$ фрагментов вычислений, выполняющих перемножение элементов, т.к. нет простого способа определить, какой элемент матрицы в каком фрагменте окажется. Если мы добавим еще один фрагмент матрицы, то число фрагментов вычислений увеличится на P. Таким образом, данный алгоритм плохо масштабируется.

На рис. 3 представлен второй способ фрагментации матрицы A, при котором внешняя структура – двумерный массив, а внутренняя – список ненулевых элементов матрицы, попавших в соответствующий блок массива. При таком способе фрагментации фрагменты матрицы оказы-

ваются разного размера. Но число фрагментов вычислений для алгоритма умножения равно числу фрагментов матрицы, поэтому данный алгоритм является хорошо масштабируемым.

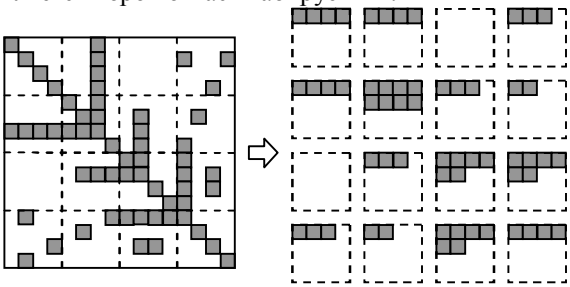


Рис. 3. Фрагментация разреженной матрицы. Способ 2

Заключение

Представлены первые результаты разработки методики фрагментации численных алгоритмов. Выделены принципы построения фрагментированных алгоритмов. Рассмотрен вопрос фрагментации регулярных структур.

Дальнейшая работа по разработке методики фрагментации связана с рассмотрением конкретных внешних и внутренних структур для реализации распространенных численных методов и алгоритмов. Кроме того, формируется список требований к исполнительной системе, которым она должна удовлетворять для обеспечения должного качества исполнения фрагментированных программ.

Литература

1. **Вальковский В.А., Малышкин В.Э.** Синтез параллельных программ и систем на вычислительных моделях. – Новосибирск: Наука, 1988. – 128 с.
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, № 6. – P. 755–765.
3. **Malyshkin V.E., Perepelkin V.A.** Optimization of Parallel Execution of Numerical Programs in LuNA Fragmented Programming System // MTPP–2010 revised selected papers. – N.Y.: Springer, LNCS 6083, 2010. – P. 1–10.

4. **Kireev S.E., Malyshkin V.E.** Fragmentation of Numerical Algorithms for Parallel Subroutines Library // The Journal of Supercomputing. – 2011. – Vol. 57, №. 2. – P. 161–171.