Представление алгоритмов в технологии фрагментированного программирования

В.А. Перепелкин
Институт вычислительной математики и математической геофизики СО РАН, Новосибирск

Рассмотрен вопрос о представлении алгоритмов в языке фрагментированного программирования с точки зрения пользователя, компилятора и исполняющего окружения. Сформулированы требования к языку и представлена неформальная модель фрагментированной программы.

Введение

Актуальной задачей является упрощение процесса создания эффективных параллельных программ [1. С. 4; 2]. Один из подходов для задач численного моделирования – это технология фрагментированного программирования [3], направленная на то, чтобы освободить программиста от низкоуровневого программирования коммуникаций, синхронизаций и распределения ресурсов во времени, а также обеспечить автоматически необходимые динамические свойства прикладной параллельной программы. Работа посвящена рассмотрению вопроса представления программ в этой технологии.

Фрагментированное программирование

Технология фрагментированного программирования предназначается для реализации сверхбольших задач численного моделирования. Она состоит в том, что параллельный алгоритм представляется как совокупность двух частей. Первая, так называемая «алгоритмическая», часть содержит представление алгоритма – данные программы, вычисления над этими данными и допустимый способ исполнения программы. Данные и вычисления агрегируются из так называемых фрагментов данных и вычислений, и такая фрагментированная структура программы сохраняется во время исполнения. Это позволяет осуществлять различные манипуляции с программой во время исполнения, например перераспределять фрагменты, планировать вычисления. Вторая часть, «рекомендательная», содержит информацию, связанную с реализацией алгоритма на конкретном вычислителе, – информацию о структуре данных и вычислений, рекомендуемый способ исполнения и

отображения программы на ресурсы суперкомпьютера. Такое представление называется фрагментированной программой (ФП).

Если алгоритм запрограммирован верно, то первой, алгоритмической, части достаточно, чтобы исполнить программу. Но эффективное ее исполнение затруднительно вследствие сложности задачи отображения программы на ресурсы параллельного суперкомпьютера во времени. Для этого требуется информация из второй, реализационной, части программы [4]. По сути, она содержит основные решения, которые требуется принять для эффективного исполнения программы.

Исполняет ФП специальное окружение, называемое исполнительной системой (ИС). ИС осуществляет отображение ФП на ресурсы суперкомпьютера и исполняет ее в соответствии с имеющимся аппаратным обеспечением и заданными рекомендациями. ИС реализует все низкоуровневые операции по управлению ресурсами, обмену данными, синхронизации и балансировке загрузки процессоров.

Создание ФП происходит на высоком уровне, слабо связанном с конкретным аппаратным окружением, жестким порядком исполнения подзадач и распределением ресурсов. Это обуславливает переносимость ФП. Для исполнения такой программы на другом классе параллельных компьютеров требуется реализовать новую ИС. Изменений в саму программу вносить не требуется, требуется только ее перекомпиляция.

Требования к представлению алгоритма

К представлению фрагментированных алгоритмов предъявляются следующие требования. Пользователь должен задавать свой алгоритм на высоком уровне, без описания низкоуровневых операций управления процессами, пересылками данных и синхронизации. Транслятор анализирует программу, оптимизирует ее, настраивает на конкретный параллельный компьютер и осуществляет предварительное планирование вычислений с учетом как структуры программы, так и рекомендаций пользователя. Представление алгоритма должно содержать всю необходимую для этого информацию. В результате транслятор генерирует исполняемую программу. Далее ИС выполняет сгенерированную программу в соответствии с составленным планом. Она осуществляет оптимизацию и принятие решений времени исполнения (распределение и перераспределение ресурсов, планирование вычислений).

ФП должна быть в значительной степени параметризована. Фрагментация данных на конкретные фрагменты данных осуществляется в нужной степени грануляции, в зависимости от имеющихся ресурсов.

Конкретные процедуры могут быть выбраны в зависимости от имеющихся библиотек и особенностей аппаратного обеспечения. Различные классы параллельных компьютеров могут использовать различные рекомендации.

Фрагментированная программа

Определим ФП как кортеж $<DF, CF, <, n, w, p, G>$.

DF – это множество фрагментов данных. Фрагмент данных – это ресурс, хранящий значения переменных алгоритма. Например, фрагмент данных может представлять некоторую область сетки в конечно-разностной схеме.

CF – это множество фрагментов вычислений. Каждый фрагмент вычислений применяется к определенным фрагментам данных и должен сработать один раз.

< – это бинарное отношение частичного порядка на множестве *CF*, которое определяет допустимый порядок срабатывания фрагментов вычислений. Если < – линейный порядок, то выполнение ФП может быть только последовательным.

Пусть $F=CF \cup DF$, а $\boldsymbol{R}$ – множество вещественных чисел.

$n: F \times F \rightarrow \boldsymbol{R}$ – функция, задающая соседство фрагментов данных и вычислений. Например, фрагменты вычислений являются соседними для фрагментов данных, которые они обрабатывают. Соседние фрагменты нужно размещать в соседних ПЭ, чтобы уменьшить накладные расходы на пересылку данных. ПЭ считаются соседними, если существует коммуникационный канал, непосредственно их соединяющий. Функция *n* задает оценку того, насколько важно размещать данную пару фрагментов рядом.

$w: F \times F \rightarrow R$ – функция веса, определяющая ресурсоемкость фрагмента. Для фрагментов данных – это их размер, а для фрагментов вычислений – оценка времени выполнения.

Функции *n* и *w* используется при планировании исполнения и динамической балансировке загрузки.

$p: F \rightarrow R$ – функция приоритета. Задает предпочтения в порядке исполнения фрагментов вычислений. Если заданный порядок < позволяет, то на исполнение первыми назначаются фрагменты вычислений с большим приоритетом (или работающие с фрагментами данных большего приоритета). Задание приоритетов позволяет направлять фронт вычислений в нужную сторону.

$G=\{ g_i \mid g_i \subset F \}$ – множество групп фрагментов. В группы объединяются фрагменты данных и вычислений, образующие относительно независимую часть программы. Информация о группах используется при планировке исполнения ФП.

Множество *DF* и частично упорядоченное множество *CF* составляют алгоритмическую часть ФП. Это тот минимум, который необходим для корректной работы программы. Остальные компоненты составляют рекомендательную часть ФП. Для одной ФП могут быть определены несколько наборов рекомендательных составляющих. Наиболее подходящий из них будет выбран при исполнении ФП.

Заключение

Представление фрагментированных алгоритмов не содержит низкоуровневых операций управления процессами, а лишь задает общий граф программы с дополнительной информацией о ее структуре. Рекомендательная информация предназначена для использования транслятором и ИС. Параметризация программы относительно размеров фрагментов данных и выбора процедур связана с конкретным синтаксическим представлением и в статье не рассматривается.

Литература

1. **Вальковский В.А., Малышкин В.Э.** Синтез параллельных программ и систем на вычислительных моделях. Новосибирск: Наука, 1988. С. 128.
2. **Kale L.V., Bohm E., Mendes C.L. et al.** Programming Petascale Applications with Charm++ and AMPI // Petascale Computing: Algorithms and Applications, CRC Press. 2008. P. 421-441.
3. **Malyshkin V.E.** Fragmented Programming of Library Parallel Numerical Subroitines // Proceedings of the 7-th Int. conference on New Trends in Software Methodologies, Tools and Techniques, IOS Press. 2007. Vol. 193. P. 425-430.
4. **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.