СТРУКТУРНОЕ ПРЕОБРАЗОВАНИЕ ГРАФА УПРАВЛЕНИЯ ПАРАЛЛЕЛЬНЫМИ ВЫЧИСЛЕНИЯМИ ДЛЯ СИСТЕМ С РАСПРЕДЕЛЁННОЙ ПАМЯТЬЮ MPI

Д.А. Попова-Коварцева

Самарский государственный аэрокосмический университет им. С.П. Королёва

Выделены формальные правила описания моделей параллельных алгоритмов, принятые в технологии графосимволического программирования. Предложены алгоритмы, реализующие эквивалентные преобразования исходной граф-модели параллельного алгоритма первоначально с помощью суперпозиции пользовательской версии модели к виду обобщенного суперагрегата. В дальнейшем, с помощью алгоритмов F-нумерации и декомпозиционного преобразования суперагрегат приводится к стандартному эквивалентному виду, способному выполняться на высокопроизводительных системах с распределённой памятью.

Введение

В настоящее время наиболее популярным способом написания параллельных программ для высокопроизводительных систем с распределенной памятью является создание параллельных кодов программ с использованием библиотек MPI или подобных им средств. Однако использование MPI связано с изменением стиля программирования и дополнительными исследованиями корректности введенного параллелизма [1]. Даже имея хорошую идею распараллеливания некоторого вычислительного процесса, исполнитель вынужден произвести фактически системный анализ разрабатываемого параллельного алгоритма и преобразовать его к виду, реализуемому соответствующими программными средствами, например, C++ и MPI.

Возникающая необходимость равнозначного владения технологиями программирования на языках высокого уровня (C++, C#) и средством распараллеливания программ MPI еще дальше отдаляет «конечных» пользователей от возможности участия в разработке авторских параллельных приложений.

В связи с чем, актуально направление исследований, связанное с автоматизацией процессов преобразования моделей параллельных алгоритмов к стандартизованному виду, пригодному для исполнения на современных суперкомпьютерах.

1. Основные положения

Существует множество графических моделей описания алгоритмов параллельных вычислений. В области представления вычислительных алгоритмов наиболее удобной является форма, близкая по форме к блок-схемам, которая реализована в параллельной версии технологии графосимволического программирования (ГСП) в виде системы PGRAPH [3, 4].

В системе PGRAPH [3] вычислительная модель параллельных алгоритмов описывается четверкой M_G=<D, A, Ψ, G>, где D – множество данных некоторой предметной области, A – множество вычислимых функций, определенных над данными предметной области, Ψ – множество дуг управления, $G = \{A, \Psi\}$ – ориентированный помеченный граф, описывающий граф-модель вычислительного процесса.

Тип дуги Ψij, исходящей из вершины i и входящей в вершину j, определим формально как функцию $T(\Psi ij) \in \{1,2,3\}$, где:

1 – последовательная дуга (описывает передачу управления на последовательных участках вычислительного процесса);

2 – параллельная дуга (обозначает начало нового параллельного вычислительного процесса);

3 – терминирующая дуга (завершает параллельный вычислительный процесс).

Для описания отдельного вычислительного процесса в ГСП вводится понятие параллельной ветви β модели - подграфа графа G, начинающегося параллельной дугой (тип этой дуги $T(\Psi ij) = 2$, на схеме помечается «кружочком») и заканчивающегося терминирующей дугой (тип этой дуги $T(\Psi ij) = 3$, на схеме помечается перечёркнутой дугой).

Формально параллельную ветвь можно определить тройкой:

$$\beta = <A_\beta, \Psi_\beta, R_\beta> ,$$

где A_β – множество вершин ветви, Ψ_β – множество дуг управления ветви, R_β – отношение над множествами вершин и дуг ветви, определяющее способ их связывания.

Графическая модель обычно содержит несколько параллельных ветвей, каждая из которых образует отдельный процесс. В этом смысле модель параллельных вычислений можно представить как объединение нескольких параллельных ветвей: $G = \bigcup \beta_k$, где β_k – параллельные ветви графа G. Каждая параллельная ветвь исполняется на отдельном процессоре.

Число параллельных ветвей в модели фиксируется при ее построении, при этом максимальное количество ветвей не ограничивается. Каждая ветвь имеет ровно один вход и один выход.

2. Декомпозиция Р-графа

Технология ГСП не налагает серьёзных ограничений на форму представления моделей параллельных алгоритмов. В частном случае, с учетом введенной выше семантики, модель параллельного алгоритма может быть представлена совокупностью иерархически вложенных двухполюсных последовательно-параллельных ориентированных P-графов, которая при «развертывании» всех вложений может быть представлена обобщенным P-графом, полюсами которого являются одна начальная (исток) и одна конечная (сток) вершины. Первоначальную последовательную ветвь агрегата, содержащую исток называют мастер-ветвью агрегата.

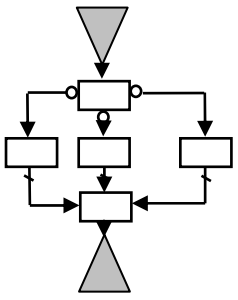

Рис. 1.

Введём понятие *правильно построенного агрегата* (ППА), схематично представленного на рис. 1. Правильно построенный агрегат имеет только одну вершину ветвления, одну терминирующую вершину и множество параллельных вершин, связанных с вершинами ветвления и терминации (см. рис.1).

Приведение обобщенной модели P-графа алгоритма к «реализационному» виду (структурное преобразование исходной модели – десуперпозиция) основывается на введении понятия F-нумерации.

В нашем случае под *F-нумерацией* графа G будем понимать инъективное отображение множества вершин $A(G)$ графа на множество слов X^* алфавита X: $F : A(G) \to X^*$.

Пусть алфавит имеет вид: $X = \{"H", "V", "E", ".", "0", "1", "2",...\}$. Слова нумерации имеют следующую семантику: $M =< N_{ur} \ . \ Type \ . \ N_1 \ . \ N_2. \ \cdots . N_{ur} >$, где N_{ur} – количество уровней иерархии при вложенности параллельных структур друг в друга; N_i – номер вершины на *i*-м уровне иерархии (если $i=N_{ur}$) или номер иерархии вложения; *Type* – тип вершины (*Type=H* для вершины ветвления; *Type=E* для терминальных вершин; для остальных *Type=V*). Представленный код вершины точно идентифицирует структуру направленного последовательно-параллельного *P*-графа.

F-нумерация начинается с начальной вершины и реализуется на основе стратегии поиска в глубину с возвратом, пока не будет реализован обход всех вершин графа [2].

На рис. 2 представлен пример модели параллельного алгоритма (модель А). Здесь вершины ветвления обозначены символом “H”. Терминирующие вершины - символом “E”. Вершины, принадлежащие параллельным ветвям, обозначены символом “#”.

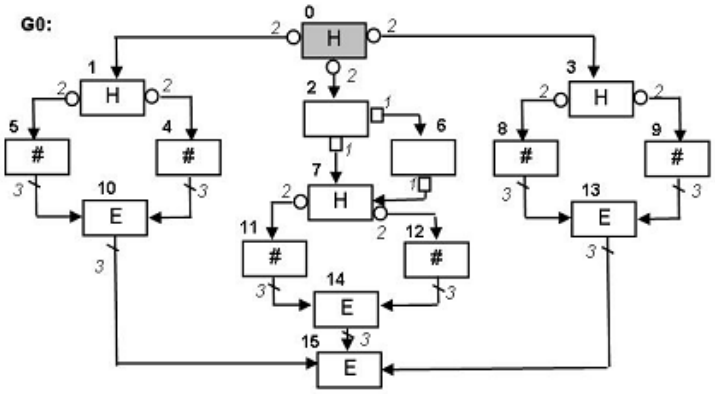

Рис. 2. Исходный *P*-граф G0 (модель А)

Мастер-ветвь агрегата G0 содержит 3 параллельные ветви, в каждую из которых вложены еще по две параллельные ветви, и в этом смысле G0 не является ППА. После реализации алгоритма разметки графа получим *F*-нумерацию вершин графа, представленную на рисунке 3.

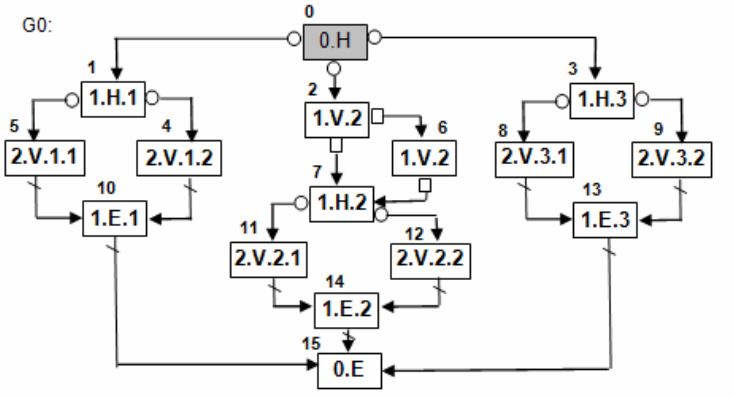

Рис. 3. Размеченный граф G0 (модель В)

Для выполнения параллельных вычислений необходимо произвести разбиение структуры графа G0 на совокупность правильно построенных агрегатов (модель С), например, как это показано на рис. 4.

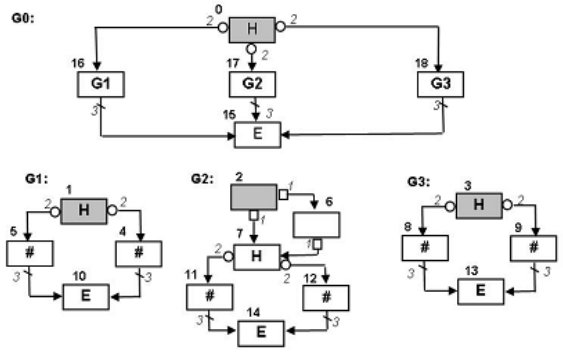

Рис. 4. Разбиение агрегата G0 на правильные компоненты (модель C)

Представленная на рис. 3 нумерация вершин графа позволяет организовать процедуру редукции (десуперпозицию) исходного *P*-графа на совокупность вложенных правильно построенных агрегатов, как это показано на рис. 4, т.е. реализовать преобразование модели A в модель C.

Десуперпозиция реализуется в соответствии со следующими правилами:

Правило 1. Вершины каждой из параллельных веток графа, находящиеся между параллельной и терминирующей вершинами одного уровня, агрегируются в отдельный подграф. Вложенному агрегату присваивается новое имя. Выделенный подграф заменяется в исходном подграфе новой вершиной.

Правило 2. Если новый подграф не является ППА, то к нему применяется правило 1.

Правило 3. Процедура десуперпозиции завершается, если для исходного и всех новых подграфов невозможно применить правило 1.

3. Преобразование графа управления модели параллельного алгоритма к стандартному (реализационному) виду

Завершающий этап приведения описания графа управления модели параллельного алгоритма к стандартизованному виду фактически сводится к удалению из модели C всех терминирующих дуг и добавлению для каждого подграфа по одной управляющей дуге, направленной от параллельной к терминальной вершине. В результате формируется стандартная форма модели параллельных вычислений (модель D), представленная на рис. 5. В модели D все концевые вершины графов (кроме терминирующей, имеющей тип E) закрепляются за одним из компьютеров кластера, т.е к процессорам p1, p2 и т.д., на которых запускаются соответствующие вычислительные задачи. После завершения всех вычислений управление передаётся на терминирующую вершину типа E.

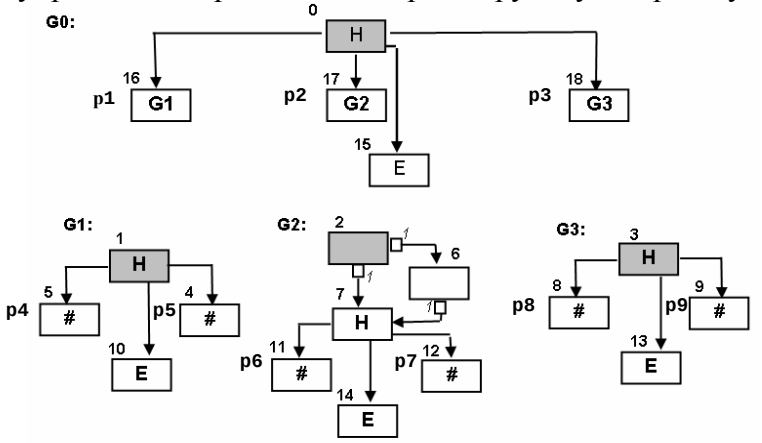

Рис. 5. Стандартная форма модели графа G0 (модель D)

В результате предложенный метод автоматизации организации параллельных вычислений для произвольной модели параллельного алгоритма сводится к последовательности эквивалентных декомпозиционных преобразований по цепочке: модель A -> модель B -> модель C -> модель D.

Заключение

Предложенный алгоритм структурного преобразования модели параллельного алгоритма позволяет без участия разработчика программы произвести все необходимые структурные преобразования графа управления модели к виду, пригодному для реализации вычислений в среде MPI.

Литература

1. Гергель В.П. Теория и практика параллельных вычислений. – М.: БИНОМ, 2007.- 423с.
2. Евстигнеев В.А. Применение теории графов в программировании. – М.: Наука, 1985. - 352 с.
3. Жидченко В.В., Коварцев А.Н. Моделирование синхронных параллельных вычислений при построении математических моделей сложных систем // Первая международная конференция «Системный анализ и информационные технологии» САИТ-2005: Труды конференции. В 2т. Т.2. – М.: КомКнига, 2005. – С. 154-160.
4. Коварцев А.Н. Автоматизация разработки и тестирования программных средств. – Самара: Самарский государственный аэрокосмический университет, 1999. – 150 с.