Вычислительные технологии, 2020, том 25, № 1, с. 66–81. © ИВТ СО РАН, 2020 ISSN 1560-7534
Computational Technologies, 2020, vol. 25, no. 1, pp. 66–81. © ICT SB RAS, 2020 eISSN 2313-691X

ВЫЧИСЛИТЕЛЬНЫЕ ТЕХНОЛОГИИ

DOI:10.25743/ICT.2020.25.1.005

К вопросу об автоматическом создании параллельных прикладных программ для реконфигурируемых вычислительных систем

И. И. Левин[1], А. И. Дордопуло[2]
[1]Южный федеральный университет, Таганрог, Россия
[2]ООО "НИЦ супер-ЭВМ и нейрокомпьютеров", Таганрог, Россия
Контактный автор: Дордопуло Алексей И., e-mail: dordopulo@superevm.ru

Поступила 5 марта 2019 г., доработана 2 октября 2019 г., принята в печать 23 октября 2019 г.

Рассмотрена оригинальная методика отображения информационного графа прикладной программы на архитектуру реконфигурируемой вычислительной системы с помощью методов редукции производительности, обеспечивающих решение задач, аппаратные затраты на реализацию которых превышают доступный вычислительный ресурс. Доказаны теоремы о свойствах последовательного применения редукций по числу базовых подграфов, по числу вычислительных устройств и разрядности. На основе доказанных теорем и следствий из них сформулирована методика редукционных преобразований информационного графа прикладной программы для автоматической адаптации к архитектуре реконфигурируемой вычислительной системы. Приведена оценка максимального числа преобразований согласно предложенной методике для сбалансированной редукции производительности и аппаратных затрат прикладных программ для реконфигурируемых вычислительных систем.

Ключевые слова: редукция производительности, аппаратные затраты, реконфигурируемая вычислительная система, разработка параллельных программ, информационный граф.

Цитирование: Левин И.И., Дордопуло А.И. К вопросу об автоматическом создании параллельных прикладных программ для реконфигурируемых вычислительных систем. Вычислительные технологии. 2020; 25(1):66–81.

Введение

Основной целью применения многопроцессорных вычислительных систем (МВС) для решения прикладных задач является сокращение времени работы параллельной программы по сравнению с последовательной в соответствии с количеством используемых для вычислений процессоров (в наилучшем случае — прямо пропорционально). Разработка параллельной программы сложна и трудоемка, потому что программисту необходимо самостоятельно организовывать и синхронизировать вычисления для большого числа процессов, выполняющихся асинхронно на процессорах (узлах) МВС. Растущие требования к сокращению времени счета и повышению точности результатов удовлетворяются, как правило, с помощью увеличения числа используемых для вычислений узлов МВС, что, в свою очередь, усложняет разработку и корректировку параллельной программы. Одним из способов создания параллельных программ для МВС различных

архитектур (с общей и распределенной памятью, в том числе кластерных вычислительных систем) является автоматическое распараллеливание последовательных программ компилятором.

На вход распараллеливающего компилятора [1] поступает императивная последовательная программа, по которой он восстанавливает естественный параллелизм исходного алгоритма и отображает его на архитектуру МВС в виде параллельной программы с некоторой эффективностью. Как правило, распараллеливающие компиляторы выявляют участки последовательной программы, которые могут быть выполнены одновременно (например, итерации циклов, которые можно выполнить параллельно), и добавляют инструкции для их параллельного выполнения.

Сложность автоматического распараллеливания для наиболее распространенных МВС с распределенной памятью — кластерных вычислительных систем состоит в том, что компилятору необходимо анализировать варианты не только распараллеливания различных фрагментов последовательной программы, но и размещения данных по процессорам с учетом характеристик коммуникационной сети. Для кластерной вычислительной системы, содержащей n узлов, в предположении, что каждый узел обрабатывает собственную локальную порцию данных, распределение данных по узлам можно представить n-арным деревом. Число вариантов размещения данных в соответствии с теоремой Кэли [2] можно оценить числом различных деревьев для n вершин, которое составляет n^{n-2}. Для 64 узлов кластерной МВС число вариантов возможных размещений данных составит $64^{62} = 2^{372}$, что не позволяет проанализировать их на современных средствах вычислительной техники в приемлемое время. Поэтому для сокращения числа деревьев, анализируемых распараллеливающим компилятором, используются эвристики [1] для сужения пространства поиска.

Для выявления параллелизма в императивной программе применяются сложнейшие методы анализа информационных зависимостей [3], гнезд и витков циклов [4], приватных и редукционных переменных, канонизации, раскрутки и развертки циклов, эвристические методы (априорный выбор последовательности оптимизационных преобразований и апостериорный подход на основе итерационной компиляции участков кода [3, 5]), которые отсекают заведомо неэффективные варианты параллельной программы, но и их использование не позволяет эффективно распараллелить последовательную программу автоматически, без рекомендаций и указаний программиста.

При программировании реконфигурируемых вычислительных систем (РВС), построенных на основе программируемых логических интегральных схем (ПЛИС), для создания параллельной программы используется альтернативный подход — задача исходно представляется в виде информационного графа [6], отражающего естественный параллелизм и конвейеризацию вычислений. Информационный граф представляет совокупность множеств входных, выходных и операционных вершин задачи, соединенных между собой информационными связями (дугами). Абсолютно параллельный информационный граф не содержит циклов, все операционные вершины задачи представлены заданное размерностью задачи число раз.

Информационный граф задачи может быть представлен в виде объединения нескольких функционально законченных подзадач, каждая из которых, в свою очередь, представима в ярусно-параллельной форме в виде множества ярусов (слоев), содержащих изоморфные друг другу параллельные фрагменты вычислений (базовые подграфы [6]). Поэтому каждая подзадача (а для многих областей — и задача) может быть

представлена в виде масштабируемого в пределах слоя или по ярусам (итерациям) хотя бы одного базового подграфа, который является минимальной структурной единицей реализации прикладной программы в РВС [6]. Реализованные в архитектуре РВС базовые подграфы информационного графа, связанные коммутационной структурой, образуют вычислительную структуру прикладной задачи, содержащую вычислительные устройства, соответствующие операционным вершинам информационного графа, которые соединены коммутационными связями, соответствующими дугам информационного графа.

Такое представление позволяет выполнять масштабирование вычислительной структуры, увеличивая число реализованных в архитектуре РВС базовых подграфов (при наличии свободного аппаратного ресурса и каналов ввода-вывода) или сокращая их число (при нехватке аппаратного ресурса). При наличии информационных зависимостей между базовыми подграфами соседних ярусов вычислительная структура может масштабироваться не только по данным, но и по итерациям [6], что существенно сокращает затраты такого критического для РВС ресурса, как каналы памяти (или ввода-вывода).

1. Использование редукции производительности и аппаратных затрат прикладных программ для реконфигурируемых вычислительных систем

Традиционные методы реализации прикладных задач на РВС [6] оперировали базовым подграфом как основной неделимой (атомарной) единицей для масштабирования вычислений. Базовый подграф был минимальной величиной для решения задачи в РВС, и считалось, что если аппаратного ресурса недостаточно для реализации базового подграфа, задачу невозможно решить на РВС. В работе [7] впервые предложено использовать методы редукции производительности прикладной задачи как инструмент сокращения аппаратных затрат в том случае, когда аппаратного ресурса РВС недостаточно для реализации даже одного базового подграфа. Производительность прикладной задачи определяется как число вычислительных операций в единицу времени при решении прикладной задачи. Общее число вычислений NC_F для обработки потока данных длины N на вычислительной структуре F, содержащей N_F базовых подграфов задачи с числом устройств Op_F каждый, обрабатывающих данные разрядности ρ_F, составит

$$NC_F = NN_F Op_F \rho_F. \tag{1}$$

Время решения прикладной задачи на вычислительном устройстве с длительностью такта $\tau = \dfrac{1}{Freq}$ и скважностью (интервалом подачи данных, выраженным в тактах) S при обработке потока данных длины N составит $t = NS\tau$. Тогда производительность прикладной задачи

$$Perf_F = \frac{NC_F}{t} = \frac{NN_F Op_F \rho_F}{NS\tau} = \frac{N_F Op_F \rho_F}{S\tau} = \frac{N_F Op_F \rho_F Freq}{S}. \tag{2}$$

Редукция производительности с целочисленным коэффициентом редукции R представляет собой сокращение производительности (2) в целое число раз обратно пропорционально значению R:

$$Perf_F(R) = \frac{Perf_F}{R} = \frac{N_F Op_F \rho_F}{S\tau R} = \frac{N_F Op_F \rho_F Freq}{SR}. \quad (3)$$

Как видно из (3), сократить производительность вычислительной структуры при решении задачи на РВС можно различными способами:

— сократив число реализованных базовых подграфов задачи N_F (традиционный способ масштабирования реализации задачи в РВС [6, 7]);

— сократив число устройств Op_F, реализующих базовый подграф задачи [7];

— сократив разрядность обрабатываемых данных ρ_F (применим для данных с фиксированной точкой и ограниченно применим для данных с плавающей точкой);

— увеличив скважность обработки данных S или сократив частоту обработки $Freq$ [7].

В первых трех случаях одновременно с сокращением производительности возможно сокращение аппаратных затрат РВС на реализацию вычислительной структуры F, если аппаратные затраты на коммутацию и синхронизацию вычислений не превысят сокращенные ресурсы. Два последних случая только сокращают производительность задачи или ее фрагмента, не оказывая влияния на аппаратные затраты, но могут использоваться для согласования темпа обработки данных в разных фрагментах задачи. Основным и наиболее важным отличием методов редукции производительности является сбалансированность получаемой в результате редукции вычислительной структуры, которая означает кратное масштабирование потоков данных и аппаратных затрат на их коммутацию и синхронизацию. Применительно к формуле (3) это означает сокращение сомножителей числителя дроби (3) на значение коэффициента редукции R (или составляющие его простые сомножители).

На первый взгляд кажется, что при редукции производительности решается задача, обратная распараллеливанию: при распараллеливании производительность последовательной программы увеличивается за счет увеличения числа одновременно работающих процессоров (устройств) и выполняющихся фрагментов программы, а при редукции производительности сокращается производительность реализации абсолютно параллельного информационного графа на РВС, при этом количество вариантов перебора остается таким же колоссальным. Но в случае редукции производительности ситуация иная: разумеется, число вариантов построения параллельной программы не меняется, но число шагов для получения итоговой параллельно-конвейерной программы для РВС существенно сокращается. При редукции производительности абсолютно параллельного информационного графа все многообразие вариантов разбивается на небольшое число классов, содержащих изоморфные варианты вычислительных структур, поэтому число вариантов перебора очень невелико. Проводя аналогию с деревом, масштабирование вычислений при редукции позволяет рассматривать только основные ветви дерева, поскольку все листовые вершины в данной ветви будут равнозначными.

Методы редукции производительности [7] использовались для реализации на РВС задач, базовые подграфы которых содержат сотни и тысячи операционных вершин в информационном графе. Это задачи молекулярного моделирования (докинг ингибиторов) и синтеза новых химических соединений, задачи трехмерного моделирования физических процессов в пространстве (томографического исследования земной поверхности с достаточным разрешением), задачи моделирования физических процессов с высокой разрешающей способностью, ряд задач символьной обработки и др. Информационные графы таких задач в абсолютно параллельной форме содержат десятки и сотни тысяч

вершин, которые невозможно реализовать на реальных РВС, поэтому их необходимо редуцировать, одновременно с производительностью сокращая занимаемый базовым подграфом аппаратный ресурс до приемлемых для реализации значений. Методы редукции производительности преобразовывали исходную вычислительную структуру таких задач в вычислительную структуру меньшего по сравнению с исходным подграфа, который будем называть модернизированным подграфом или м-подграфом. М-подграф при реализации на РВС позволяет на меньшем (по сравнению с реализацией исходного базового подграфа) числе вычислительных устройств выполнить все вычисления базового подграфа с пропорциональным увеличением времени решения.

Не останавливаясь на механизме и деталях реализации редукционных преобразований (редукций), подробно изложенных в [7], в качестве основных редукций будем рассматривать следующие:

— редукцию по числу базовых подграфов R^N, сокращающую число одновременно реализованных в РВС базовых подграфов;

— редукцию по числу устройств R^{Op}, сокращающую число одновременно выполняемых операций базового подграфа за счет совмещения одинаковых операций базового подграфа с одинаковыми типами данных в одном устройстве с синтезом коммутационной структуры для синхронизации операндов;

— редукцию по разрядности R^{ρ}, сокращающую число одновременно обрабатываемых разрядов за счет преобразования параллельной обработки разрядов каждого операнда в параллельно-последовательную (в предельном случае — в последовательную) обработку.

Все рассматриваемые редукции независимы, поскольку влияют на ортогональные характеристики информационного графа задачи.

Редукция по числу базовых подграфов R^N [7] позволяет достаточно просто и с линейным коэффициентом сократить число базовых подграфов в информационном графе задачи и реализовать ее на РВС при условии, что хотя бы один базовый подграф возможно реализовать в архитектуре вычислительной системы. Если базовый подграф по аппаратному ресурсу или числу каналов памяти не может быть реализован в РВС, необходимо сокращать аппаратные затраты на реализацию критического ресурса до допустимого значения с помощью редукции по числу устройств R^{Op} и/или редукции по разрядности R^{ρ}.

2. Применение редукционных преобразований для синтеза вычислительной структуры прикладной программы

Определим взаимное влияние последовательно применяемых редукций на синтезируемую вычислительную структуру м-подграфа, на значение коэффициента редукции и число шагов, необходимых для синтеза эффективной масштабируемой вычислительной структуры при решении прикладной задачи на РВС. Последовательное применение редукций будем обозначать знаком $\times$, например, последовательное применение редукций по числу базовых подграфов и по числу устройств будет иметь вид $R^N \times R^{Op}$.

Как уже отмечалось, при редукции производительности с целью сокращения аппаратных затрат выполняется сбалансированное сокращение числа базовых подграфов, вычислительных устройств и разрядности обрабатываемых данных в заданное натуральным коэффициентом редукции число раз до целого значения, которое не может

быть меньше единицы. Для описания такого сокращения будем использовать операцию округления в меньшую сторону [8] до единичного значения, которую для натуральных чисел $a \geq 1, b \geq 1$ определим как

$$\left\lfloor \frac{a}{b} \right\rfloor_1 = \begin{cases} a \ \mathbf{div} \ b, & a > b, \\ 1, & a \leq b, \end{cases} \tag{4}$$

где **div** — операция целочисленного деления, а нотация $\lfloor \ \rfloor_1$, выбранная по аналогии с нотацией $\lfloor \ \rfloor$ ("пол") и [8], указывает, что значение операции "пол" ограничено снизу единичным значением.

Результат операции округления в меньшую сторону, заданный выражением (4), соответствует физическому смыслу редуцируемых параметров, потому что число базовых подграфов, вычислительных устройств и обрабатываемых разрядов данных после редукции не может быть меньше 1. Заданная выражением (4) операция округления обладает полезным свойством, приведенным в [8], — для любых вещественных m, x и натурального числа n выполняется следующее равенство:

$$\left\lfloor \frac{\left\lfloor \frac{x}{m} \right\rfloor}{n} \right\rfloor = \left\lfloor \frac{x}{mn} \right\rfloor. \tag{5}$$

Поскольку натуральные числа — подмножество вещественных чисел, а функция $\left\lfloor \frac{a}{b} \right\rfloor_1$ является монотонной и непрерывной, равенство (5) справедливо и для нее, поэтому с учетом коммутативности

$$\left\lfloor \frac{\left\lfloor \frac{x}{m} \right\rfloor_1}{n} \right\rfloor_1 = \left\lfloor \frac{\left\lfloor \frac{x}{n} \right\rfloor_1}{m} \right\rfloor_1 = \left\lfloor \frac{x}{mn} \right\rfloor_1. \tag{6}$$

С помощью формулы (6) докажем важную для последовательных редукций теорему о представлении коэффициента редукции в виде произведения коэффициентов последовательных редукционных преобразований.

Теорема 1. *Последовательно примененные редукции R_m^T и R_n^T одного типа T с натуральными коэффициентами $m > 1$ и $n > 1$ эквивалентны редукции R_{nm}^T того же типа с коэффициентом, равным их произведению $(mn) > 1$:*

$$R_n^T \times R_m^T = R_{nm}^T. \tag{7}$$

Доказательство. Будем рассматривать фрагмент F задачи, содержащий N_F базовых подграфов, с числом устройств Op_F каждый, обрабатывающих данные разрядности ρ_F. Общий объем вычислений в фрагменте F равен

$$NC_F = N_F Op_F \rho_F.$$

Поскольку редукционные преобразования независимы, условие (7) теоремы 1 необходимо доказать для каждого вида редукции (по числу базовых подграфов, числу вычислительных устройств, разрядности).

Докажем условие (7) для редукции по числу базовых подграфов R_n^N с коэффициентом n. Число базовых подграфов фрагмента F задачи сократится до величины $\left\lfloor \frac{N_F}{n} \right\rfloor_1$, а общий объем вычислений составит

$$NC_n^N = \left\lfloor \frac{N_F}{n} \right\rfloor_1 Op_F \rho_F. \tag{8}$$

Последовательная редукция этого же фрагмента R_n^N сократит число базовых подграфов в m раз, и формула (8) с учетом (6) примет вид

$$NC_{n\times m}^{N\times N} = \left\lfloor \frac{\left\lfloor \frac{N_F}{n} \right\rfloor_1}{m} \right\rfloor_1 Op_F \rho_F = \left\lfloor \frac{N_F}{nm} \right\rfloor_1 Op_F \rho_F. \tag{9}$$

Общее число вычислений при последовательной редукции по числу базовых подграфов с коэффициентами n и m из выражения (9) в точности соответствует результату редукции R_{nm}^N фрагмента F задачи по числу базовых подграфов при подстановке в формулу (8) коэффициента nm вместо n, что доказывает теорему 1 для редукции фрагмента задачи по числу базовых подграфов. Аналогичным образом доказывается условие (7) для редукции R_n^{Op} по числу операций Op и редукции R_n^ρ по разрядности обрабатываемых данных ρ, что доказывает теорему 1 в целом. ■

На основании теоремы 1 можно сформулировать важные для практического применения редукционных преобразований следствия.

Следствие 1.1. *Представление коэффициента редукции производительности в виде произведения простых сомножителей позволяет сократить число шагов для подбора рациональных значений коэффициентов последовательной редукции одного или разных типов. Разложение коэффициента редукции на простые сомножители позволяет выбрать наиболее рациональное значение коэффициента для редукции каждого типа, и наоборот — для заданного значения коэффициента редукции возможно определить наиболее подходящий тип редукции с учетом характеристик задачи, представленной формулой (1). Если коэффициент редукции производительности R является простым числом, большим 2, и значение R недостижимо с помощью однократной редукции, целесообразно осуществлять редукцию не в R, а в $(R+1)$ раз, чтобы сократить аппаратные затраты не менее чем в R раз. Поскольку $(R+1)$ в этом случае будет четным составным числом, возможно последовательное применение редукционных преобразований с коэффициентами редукции из разложения числа $(R+1)$ на простые сомножители.*

Следствие 1.2. *При кратном увеличении коэффициента редукции нет необходимости возвращаться к исходному базовому подграфу при последовательном применении редукции одного типа. Действительно, если полученная в результате редукции вычислительная структура нуждается в дополнительном кратном (не менее чем вдвое) сокращении аппаратных затрат и для данного типа редукции возможно кратное увеличение ее коэффициента, то в соответствии с теоремой 1 последовательно примененная редукция без возврата к исходному базовому подграфу сократит число шагов для получения итоговой редуцированной структуры.*

Согласно теореме 1 и следствиям 1.1 и 1.2 при последовательно примененных редукциях суммарный коэффициент равен произведению (а не алгебраической сумме)

коэффициентов редукций. Поэтому из редуцированной с коэффициентом n вычислительной структуры с помощью последовательно примененной редукции любого типа нельзя получить редуцированную с коэффициентом $n+1$ вычислительную структуру. Докажем это утверждение более строго в общем виде для коэффициентов вида $(n+x)$ с помощью теоремы 2.

Теорема 2. *Для редуцированного с коэффициентом n базового подграфа с помощью последовательного применения дополнительной редукции любого типа T с любым натуральным коэффициентом $k > 1$ в общем случае невозможно получить вычислительную структуру с коэффициентом редукции $(n+x)$ для любого наперед заданного x* $\forall x \geq 1$:

$$R_n^{T1} \times R_k^{T2} \neq R_{n+x}^{T}. \tag{10}$$

Доказательство. В соответствии с теоремой 1 для редукций одного типа T условие (10) будет выполняться только в случае равенства коэффициентов редукции

$$nk = n + x.$$

Тогда

$$k = 1 + \frac{x}{n}. \tag{11}$$

Поскольку по условиям теоремы 2 числа n, k, x — натуральные, уравнение (11) имеет решения для k в области натуральных чисел только для случая, когда x нацело делится на n, а не для $\forall x \geq 1$, что доказывает теорему 2 для редукции одного типа T. ■

Для редукции разных типов у одинаковых вычислительных структур из левой и правой частей (10) должно быть одинаковое общее число вычислительных операций

$$NC_{n\times k}^{T1\times T2} = NC_{n+k}^{T},$$

поэтому

$$\left\lfloor \frac{NC}{nk} \right\rfloor_1 = \left\lfloor \frac{NC}{n+x} \right\rfloor_1. \tag{12}$$

Для выполнения условия (12) необходимо, чтобы

$$nk \geq n + x \quad \text{и} \quad n + x \geq nk.$$

Одновременное выполнение этих условий возможно только в случае

$$nk = n + x. \tag{13}$$

Тогда

$$k = 1 + \frac{x}{n}. \tag{14}$$

Поскольку по условиям теоремы 2 числа n, k, x — натуральные, уравнение (14) имеет решения для k в области натуральных чисел только для случая, когда x нацело делится на n, а не для $\forall x \geq 1$, что приводит к противоречию и тем самым доказывает теорему 2.

На основании теоремы 2 можно сформулировать важное для определения последовательности применения редукционных преобразований следствие.

Следствие 2.1. *Поскольку при последовательных редукциях любых типов невозможно увеличить коэффициент редукции на произвольную величину для уже редуцированной структуры, в общем случае при необходимости дополнительной редукции (дополнительного сокращения аппаратных затрат) необходимо возвращаться к исходному*

базовому подграфу и проводить редукционные преобразования с новым (увеличенным) значением коэффициента редукции R. Такой возврат к исходному базовому подграфу увеличивает число шагов для получения редуцированной структуры.

Рассмотрим влияние последовательности применения редукции разных типов на итоговую вычислительную структуру.

Теорема 3. *Суперпозиция редукций различных типов ($T1$ с коэффициентом n и $T2$ с коэффициентом m) коммутативна — изменение порядка выполнения редукций различных типов $T1$ и $T2$ в последовательных редукционных преобразованиях фрагмента задачи не изменяет результирующий информационный граф м-подграфа:*

$$R_n^{T1} \times R_m^{T2} = R_m^{T2} \times R_n^{T1},$$

где $T1, T2$ — типы редукционных преобразований; n, m — коэффициенты редукции.

Доказательство. Докажем коммутативность последовательной редукции по числу базовых подграфов и числу устройств:

$$R_n^N \times R_m^{Op} = R_m^{Op} \times R_n^N.$$

После редукции R_n^N фрагмента задачи по числу базовых подграфов с коэффициентом n общее число вычислений над двоичными разрядами NC_n^N составит

$$NC_n^N = \left\lfloor \frac{N_F}{n} \right\rfloor_1 Op_F \rho_F.$$

Поскольку число базовых подграфов и число вычислительных устройств в базовом подграфе — независимые величины, последовательная редукция R_m^{Op} фрагмента задачи по числу вычислительных устройств с коэффициентом m сократит только число устройств и общее число вычислений над двоичными разрядами составит

$$NC_{n\times m}^{N\times Op} = \left\lfloor \frac{N_F}{n} \right\rfloor_1 \left\lfloor \frac{Op_F}{m} \right\rfloor_1 \rho_F.$$

Для правой части равенства (13) последовательные редукции R_m^{Op} и R_n^N этого фрагмента задачи приведут к такому же общему числу вычислений над двоичными разрядами

$$NC_{m\times n}^{Op\times N} = \left\lfloor \frac{N_F}{n} \right\rfloor_1 \left\lfloor \frac{Op_F}{m} \right\rfloor_1 \rho_F.$$

Аналогичным образом доказывается коммутативность для всех остальных возможных сочетаний последовательных редукций разных типов, что доказывает теорему 3. ■

На основании теоремы можно сформулировать важное для определения числа шагов редукционных преобразований следствие.

Следствие 3.1. *При изменении порядка выполнения редукционных преобразований необязательно возвращаться к исходному базовому подграфу для сокращения числа шагов.*

3. Методика отображения информационного графа прикладной программы на архитектуру реконфигурируемой вычислительной системы с помощью методов редукции производительности

На основании доказанных теорем 1–3 и следствий из них можно сформулировать основные принципы методики редукционных преобразований информационного графа задачи для адаптации к архитектуре гибридной реконфигурируемой вычислительной системы.

1. Если число базовых подграфов в информационном графе задачи больше 1, то первым шагом редукционных преобразований целесообразно выполнять редукцию по числу базовых подграфов.

Редукция производительности по числу базовых подграфов линейно сокращает занимаемый аппаратный ресурс как по числу логических ячеек ПЛИС, так и по числу каналов (при распараллеливании по данным) и пропорционально коэффициенту редукции линейно увеличивает время решения задачи. Редукция по числу базовых подграфов не требует дополнительных коммутационных ресурсов и преобразований вычислительной структуры базового подграфа задачи (по числу или разрядности вычислительных устройств). Поэтому для наибольшего сокращения аппаратных затрат без дополнительных накладных расходов целесообразно в первую очередь использовать редукцию производительности по числу базовых подграфов.

2. При выполнении редукции по числу вычислительных устройств и разрядности для сокращения числа шагов редукционных преобразований целесообразно проводить редукцию каждого типа до максимального значения, выбранного с учетом сомножителей коэффициента редукции информационного графа или типа редукции, после чего переходить к редукции другого типа.

Редукция по числу вычислительных устройств и разрядности может приводить к дополнительным накладным аппаратным затратам на коммутационное оборудование, что может увеличить аппаратные затраты на реализацию редуцированной вычислительной структуры таким образом, что они не будут удовлетворять заданному коэффициентом редукции значению и требовать дополнительного шага редукционных преобразований для сокращения аппаратных затрат. Поэтому целесообразно провести редукцию по числу вычислительных устройств и разрядности до предельных значений, чтобы определить величину дополнительных аппаратных затрат на коммутацию и уточнить значение коэффициента редукции для следующей редукции.

3. Для сокращения числа шагов редукционных преобразований целесообразно выбирать коэффициенты редукции каждого типа из разложения коэффициента редукции на простые сомножители.

Основной алгоритмической проблемой редукционных преобразований является выбор наиболее рационального варианта сочетания коэффициентов редукции, обеспечивающего высокую производительность при заданном значении аппаратных затрат из всего многообразия возможных значений коэффициентов редукции разных типов (по числу базовых подграфов, числу вычислительных устройств и разрядности). Сокращение этих параметров в целое число раз, пропорциональное сомножителям коэффициента редукции R, позволяет получить сбалансированную вычислительную структуру без непродуктивных аппаратных затрат на коммутацию и синхронизацию потоков дан-

ных и сократить число шагов редукции для получения эффективной масштабируемой вычислительной структуры при решении прикладной задачи на РВС.

Оценим число шагов редукционных преобразований для адаптации информационного графа задачи к архитектуре гибридной РВС, содержащей помимо ПЛИС узлы с традиционными процессорами.

Рассмотрим наиболее общий случай, когда для информационного графа задачи необходимо выполнить все рассмотренные редукционные преобразования: по числу базовых подграфов, числу устройств и разрядности обрабатываемых данных. Начальное (приблизительное) значение коэффициента редукции производительности R^0 можно получить как округленное к ближайшему целому вверх значение коэффициента сокращения аппаратных затрат K_{ann}, равного отношению аппаратного ресурса, необходимого для реализации прикладной задачи HC_{task}, к доступному аппаратному ресурсу РВС HC_{RCS}:

$$R^0 = \lceil K_{ann} \rceil = \left\lceil \frac{HC_{task}}{HC_{RCS}} \right\rceil.$$

Редукция производительности проводится для значений $R^0 > 1$, которые в соответствии со следствиями 1.1 и 3.1 представляются в виде произведения простых сомножителей:

$$R^0 = \prod_i r_i^0. \tag{15}$$

В соответствии с основной теоремой арифметики любое натуральное большее 1 число либо является простым, либо представимо в виде произведения простых сомножителей, причем единственным образом, поэтому представление (15) всегда возможно.

Для выполнения редукций с учетом разложения коэффициента редукции R_0 на простые множители он представляется в виде произведения трех коэффициентов редукционных преобразований с учетом параметров задачи из формулы (1):

$$R^0 = R_N^0 R_{Op}^0 R_\rho^0. \tag{16}$$

Поскольку в рассматриваемом случае выполняются все редукции, все коэффициенты редукции: R_N^0 (по числу базовых подграфов), R_{Op}^0 (по числу вычислительных устройств), R_ρ^0 (по разрядности данных) — должны быть больше 1. Если коэффициент редукции R^0 — простое число и он не представим с учетом параметров задачи в виде (16), то в соответствии со следствием 1.1 значения коэффициентов редукций в формуле (16) должны быть рассчитаны для значения $(R^0 + 1)$.

Согласно первому принципу сформулированной методики на первом шаге будет выполнена редукция производительности по числу базовых подграфов с коэффициентом R_N^0, на втором шаге — по числу вычислительных устройств базового подграфа с коэффициентом R_{Op}^0. В общем случае при сокращении числа устройств для реализации базового подграфа можно синтезировать не менее пяти различных м-подграфов, каждый из которых будет характеризоваться своей скважностью и аппаратными затратами:

1. М-подграф μ_1 — минимальный м-подграф, образованный опорным множеством **Supp (Op)** [9], т. е. каждая вычислительная операция базового подграфа включается не более одного раза в м-подграф.

2. М-подграф μ_2 — м-подграф, полученный при кратном целочисленном сокращении числа всех операций базового подграфа в одинаковое число раз (если это возможно),

что соответствует вынесению общего множителя для кратностей операций каждого типа (например, для базовой операции быстрого преобразования Фурье).

3. М-подграф μ_3 — м-подграф, образованный максимальным по суммарному числу операций ярусом ярусно-параллельной формы базового подграфа с дополнением до опорного множества **Supp (Op)** (при необходимости) — каждая вычислительная операция базового подграфа включается не менее одного раза в м-подграф, а ярус, содержащий наибольшее число операций, включается полностью и будет реализован за 1 такт.

4. М-подграф μ_4 — м-подграф, образованный максимальным по числу различных операций ярусом ярусно-параллельной формы базового подграфа с дополнением до опорного множества **Supp (Op)** (при необходимости) — каждая вычислительная операция базового подграфа включается не менее одного раза в м-подграф, а ярус, содержащий наибольшее число различных операций, включается полностью и будет реализован за 1 такт.

5. М-подграф μ_5 — минимальный м-подграф μ_1 с одним дополнительным устройством, реализующим наиболее используемую в базовом подграфе операцию, что примерно вдвое сокращает скважность реализации при незначительных затратах вычислительных ресурсов.

На третьем шаге преобразований для каждого полученного после редукции по числу вычислительных устройств м-подграфа будет выполнена редукция по разрядности обрабатываемых данных с коэффициентом R^0_ρ, при этом число возможных вариантов редукции по разрядности для возможных типов данных не превышает двух:

— при редукции по разрядности логических и целочисленных данных (данных с фиксированной точкой) сокращение аппаратных затрат линейно пропорционально коэффициенту редукции, поэтому редукция будет выполнена в заданное коэффициентом число раз, не превышающее разрядности обрабатываемых данных;

— при редукции данных в формате с плавающей точкой число возможных значений коэффициентов редукции для 32-разрядных данных равно одному (двукратная редукция по разрядности), а для 64-разрядных данных равно двум (двух- и четырехкратная редукция по разрядности), что обусловлено экспоненциальным ростом накладных расходов на обработку мантиссы и порядка для других значений коэффициентов редукции по разрядности.

Таким образом, после третьего шага редукционных преобразований число вариантов м-подграфов не превысит $5 \cdot 2 = 10$, для каждого из которых необходимо проанализировать занимаемый аппаратный ресурс и скважность обработки данных, определяющую время решения прикладной задачи. В ряде случаев, если аппаратные затраты итоговой редуцированной структуры задачи HC^1_{task} превышают доступный аппаратный ресурс РВС HC_{RCS}, может потребоваться дополнительный (четвертый) шаг преобразований. Такая ситуация может возникнуть из-за появления дополнительных коммутационных расходов при редукциях по числу вычислительных устройств и по разрядности для данных в формате с плавающей точкой, поскольку сокращение аппаратных затрат в этих случаях нелинейно.

При редукции по числу вычислительных устройств точно определить коэффициент редукции R^0_{Op} до выполнения преобразования возможно далеко не всегда, поэтому значения коэффициентов R^0_{Op} и R^0_ρ могут потребовать коррекции после редукции.

На четвертом шаге либо происходит возврат к исходному базовому подграфу (в соответствии с теоремой 2) и редукция производительности (шаги 1–3) с увеличенным

коэффициентом $R^1 = R^0 + 1$, либо, если это возможно, выполняется кратная редукция в два или более раз по одному из параметров для десяти рассматриваемых вариантов. Несложно заметить, что в обоих случаях число проанализированных вариантов удвоится и составит 20. Более строго, число анализируемых вариантов реализации м-подграфов можно оценить как $2^2 N(\mu)$, где $N(\mu)$ определяет число различных вариантов построения м-подграфа при редукции по устройствам.

Рассмотренные в статье варианты формирования м-подграфа при редукции по устройствам разработаны для задач цифровой обработки сигналов, линейной алгебры и докинга, и, конечно, этот список не является исчерпывающим. Вполне возможно предложить другие варианты формирования м-подграфа, которые будут эффективны для задач других классов, но вряд ли число всех способов превысит 16 ($N(\mu) = 2^4$). Даже с учетом наличия в структуре задачи нескольких фрагментов число вариантов их согласования с помощью вспомогательных редукционных преобразований (изменение частоты работы и скважности обработки данных) и схем организации обработки данных (параллельной, конвейерной, макроконвейерной, конвейера конвейеров) и число рассматриваемых вариантов не превысят 2^6 даже для задач, содержащих 8–10 различных базовых подграфов. На практике большинство прикладных задач содержат не более 3–5 различных фрагментов, поэтому общее число вариантов редукционных преобразований таких задач редко превышает 12.

Заключение

Полученная оценка числа вариантов синтезируемой при редукции производительности и аппаратных затрат вычислительной структуры существенно меньше приведенного ранее числа вариантов распределения данных для МВС с распределенной памятью, что объясняется разбиением всего множества вариантов на топологически изоморфные группы решений при редукции. Редукция сама по себе никоим образом не меняет общее число вариантов, но за счет применения базирующейся на теоремах 1–3 и следствиях из них методики позволяет разбить множество вариантов на несколько анализируемых классов, для каждого из которых достаточно проанализировать один вариант, а не все множество. Сокращение числа анализируемых вариантов до одного варианта из каждого класса очень существенно уменьшает время создания адаптированной под архитектуру и конфигурацию РВС вычислительной структуры параллельной программы.

На основе доказанных теорем и сформулированных принципов методики выполнения редукционных преобразований создан прототип компилятора для гибридных реконфигурируемых вычислительных систем [10], с помощью которого в настоящее время ведется отладка прикладных программ математической физики: решения систем линейных алгебраических уравнений (СЛАУ) методом Гаусса, решения СЛАУ методом Якоби, решения СЛАУ с помощью верхней и нижней треугольных матриц (LU-разложение). Число шагов для всех перечисленных задач, рассчитанное в соответствии с предложенной методикой редукционных преобразований, не превышает 12. Полученные значения коэффициентов редукции, числа шагов преобразований и практические результаты при масштабировании указанных задач подтверждают корректность и работоспособность описанных в настоящей статье методов редукционных преобразований для автоматического создания параллельных прикладных программ для реконфигурируемых вычислительных систем, обеспечивая при этом эффективность полученных решений не ниже 50–75 % от решений этих же задач, полученных специалистами-схемотехниками.

Список литературы / References

[1] **Клинов М.С., Крюков В.А.** Автоматическое распараллеливание Фортран-программ. Отображение на кластер. Тр. XI Всерос. науч. конф. "Научный сервис в сети Интернет", Новороссийск, 2009. М.: ИПМ им. М.В. Келдыша РАН; 2009:227–237.

[2] **Овчинников В.А.** Графы в задачах анализа и синтеза структур сложных систем. М.: МГТУ им. Н.Э. Баумана; 2014: 423.

[3] **Бахтин В.А., Жукова О.Ф., Катаев Н.А. и др.** Автоматизация распараллеливания программных комплексов. Тр. XI Всерос. науч. конф. "Научный сервис в сети Интернет", Новороссийск, 2016. М.: ИПМ им. М.В. Келдыша РАН; 2016: 76–85.

[4] Система автоматизированной параллелизации ФОРтран-программ (САПФОР). Адрес доступа: `http://www.keldysh.ru/dvm/SAPFOR/` (дата обращения 26.02.2019).

[5] **Штейнберг Б.Я.** Математические методы распараллеливания рекуррентных циклов для суперкомпьютеров с параллельной памятью. Ростов-на-Дону: Изд-во Ростовского гос. ун-та; 2004: 172.

[6] **Каляев А.В., Левин И.И.** Модульно-наращиваемые многопроцессорные системы со структурно-процедурной организацией вычислений. М.: Янус-К; 2003: 380.

[7] **Левин И.И., Сорокин Д.А., Мельников А.К., Дордопуло А.И.** Решение задач с существенно-переменной интенсивностью потоков данных на реконфигурируемых вычислительных системах. Вестн. компьютерных и информ. технологий. 2012; (2): 49–56.

[8] **Кнут Д., Грэхем Р., Паташник О.** Конкретная математика. Основание информатики. 2-е изд. М.: Мир; Бином. Лаборатория знаний; 1998: 703.

[9] **Петровский А.Б.** Пространства множеств и мультимножеств. М.: Едиториал УРСС; 2003: 248.

[10] **Левин И.И., Дордопуло А.И., Гудков В.А. и др.** Средства программирования реконфигурируемых и гибридных вычислительных систем на основе ПЛИС. XIII междунар. конф. "Параллельные вычислительные технологии" (ПаВТ-2019), короткие статьи и описания плакатов. Челябинск: Изд. центр ЮУрГУ; 2019: 477.

Вычислительные технологии, 2020, том 25, № 1, с. 66–81. © ИВТ СО РАН, 2020 ISSN 1560-7534
Computational Technologies, 2020, vol. 25, no. 1, pp. 66–81. © ICT SB RAS, 2020 eISSN 2313-691X

COMPUTATIONAL TECHNOLOGIES

DOI:10.25743/ICT.2020.25.1.005

On the problem of automatic development of parallel applications for reconfigurable computer systems

Levin Il'ya I.[1], Dordopulo Alexey I.[2]
[1]Southern Federal University, 347900, Taganrog, Russia
[2]Supercomputers and Neurocomputers Research Center, 347900, Taganrog, Russia
Corresponding author: Dordopulo, Alexey I., e-mail: `dordopulo@superevm.ru`
Received March 3, 2019, revised October 2, 2019, accepted October 23, 2019

Abstract

To solve applied problems, the hardware costs of which exceed the available computing resource of FPGA-based computer systems, an original technique was developed for mapping the informa-

tion graph of an application program to the architecture of a reconfigurable computing system. The proposed technique is based on the performance reduction methods that reduce the productivity of an applied task, which, along with the reducing productivity, does so for the hardware costs of its implementation and, thereby, solve the problem on the available computing resource. We demonstrate that the decrease in hardware costs for the computing structure realization occurs only during the reduction the basic subgraph number, the number of computing devices in a basic subgraph and the data width. The influence of sequential reduction transformations on the computing structure of a problem is examined. The proved theorems are concerned with the possibility of representing the reduction coefficient as a product of the coefficients of successive reductions, on the inability of additive increase in reduction coefficient during sequential reductions and on the superposition commutativity of different sequential reductions. The proved theorems and the corollaries presented in the article allow formulating the basic principles for the method of reduction transformations of the information graph of the problem for adaptation to the architecture of a hybrid reconfigurable computing system. A distinctive feature of the technique is a relatively small number of transformations for a balanced reduction of the information graph of the problem and the implementation of the task on a reconfigurable computer system.The comparatively small number of transformations required for the balanced reduction of the information graph of the problem and for the implementation of calculations on a reconfigurable computer system is the distinctive feature of the technique. For the developed technique, we estimated the maximal number of transformations and found out the decrease in the quantity of analyzed reduction variants from each class. The proposed technique permits the significant reduction of the time needed to create the computational structure of a parallel program adapted to the architecture and configuration of the reconfigurable computing system. Furthermore, the technique allows automatization of this process using the specialized software and providing at least 50–75 % efficiency in comparison with the solutions of the same problems by specialists.

Keywords: performance reduction, hardware costs, reconfigurable computer system, parallel applications development, information graph.

Citation: Levin I.I., Dordopulo A.I. On the problem of automatic development of parallel applications for reconfigurable computer systems. Computational Technologies. 2020; 25(1):66–81. (In Russ.)

References

1. Klinov M.C., Kryukov V.A. Automatic parallelization of Fortran programs. Mapping on cluster. Proc. of the XI All-Russ. Conf. “Scientific Service on the Internet”, Novorossiysk, 2009. Moscow: IPM im. Keldysha RAN; 2009:227–237. (In Russ.)
2. Ovchinnikov V.A. Graphs in the problems for analysis and synthesis of complex systems structures. Moscow: MGTU. im. N.E. Baumana; 2014: 423. (In Russ.)
3. Bakhtin V.A., Zhukova O.F., Kataev N.A. et al. Automation of parallelization of software systems. Proc. of the XI All-Russ. Conf. “Scientific Service on the Internet”, Novorossiysk, 2009. Moscow: IPM im. M.V. Keldysha RAN; 2016: 76–85. (In Russ.)
4. System for automated parallelization of FORtran programs (SAPFOR). Available at: `http://www.keldysh.ru/dvm/SAPFOR/` (accessed 26.02.2019). (In Russ.)
5. Shteynberg B.Ya. Mathematical methods of parallelization of recurrent cycles for supercomputers with parallel memory. Rostov-on-Don: Rostov State University; 2004: 172. (In Russ.)
6. Kalyaev A.V., Levin I.I. Modular-scalable multiprocessor computer systems with structural-procedural organization of calculations. Moscow: Yanus-K; 2003: 380. (In Russ.)
7. Levin I.I., Sorokin D.A., Melnikov A.K., Dordopulo A.I. Solving tasks with considerably variable data flow density on reconfigurable computer systems. Herald of Computer and Information Technologies. 2012; (2):49–56. (In Russ.)
8. Graham R.L., Knuth D.E., Patashnik O. Concrete mathematics. A foundation for computer science. Second Edition. United States of America: Addison-Wesley Publishing Company, Inc.; 1994: 670.

9. Petrovskiy A.B. Spaces of sets and multisets. Moscow: Editorial URSS; 2003: 248. (In Russ.)
10. Levin I.I., Dordopulo A.I., Gudkov V.A. et al. Programming tools for FPGA-based reconfigurable and hybrid computer systems. XIII international conference "Parallel computational technologies 2019" (PCT-2019), short articles and descriptions of posters. Chelyabinsk: Izdatel'skiy Tsentr YuUrGU; 2019: 477. (In Russ.)