Ресурсонезависимое программирование вычислительных систем гибридного типа на языке программирования COLAMO*

А.И. Дордопуло[1], И.И. Левин[2], И.А. Каляев[1], В.А. Гудков[2], А.А. Гуленок[1]

[1] Научно-исследовательский институт многопроцессорных вычислительных систем Южного федерального университета, г. Таганрог, Россия
[2] Научно-исследовательский центр супер-ЭВМ и нейрокомпьютеров, г. Таганрог, Россия

В статье рассматриваются методы программирования вычислительных систем гибридного типа, содержащих реконфигурируемые и микропроцессорные вычислительные узлы. Технология ресурсонезависимого программирования позволяет описывать в единой параллельно-конвейерной форме на языке программирования высокого уровня COLAMO различные виды параллельных вычислений – структурную, структурно-процедурную, мультипроцедурную и процедурную формы организации вычислений. Преобразования формы организации вычислений в программе выполняются автоматизированно препроцессором языка COLAMO с помощью методов редукции производительности.

Ключевые слова: редукция производительности, язык программирования высокого уровня, программирование вычислительных систем гибридного типа, технология ресурсонезависимого программирования.

1. Введение

Большинство реальных практических задач, решаемых на современных высокопроизводительных вычислительных системах, требует совмещения в едином вычислительном контуре как последовательных, так и параллельных вычислительных фрагментов для эффективной реализации структурных [1] и процедурных [1] фрагментов вычислений. Решение этой проблемы многие разработчики видят в создании вычислительных систем с гибридной организацией вычислений, содержащих различные по архитектуре вычислительные узлы, объединенные каналами передачи данных и позволяющие реализовать структурные и процедурные вычисления в едином вычислительном контуре. Симбиоз узлов различной архитектуры в одной вычислительной системе теоретически позволяет повысить реальную производительность вычислительной системы за счет возможности эффективной реализации как структурных, так и процедурных фрагментов вычислений на узлах различной архитектуры.

Широкое применение таких вычислительных систем для решения практических прикладных задач существенно ограничивается высокой сложностью их программирования, поскольку для эффективного использования архитектурных преимуществ всех вычислительных узлов программисту необходимо не только хорошо знать разные языки программирования и среды разработки для вычислительных узлов различных типов, но и самостоятельно синхронизировать вычисления в едином контуре.

2. Программирование вычислительных систем гибридного типа

Вычислительная система гибридного типа (ВСГТ) содержит различные по архитектуре вычислительные узлы с различным типом организации вычислений. ВСГТ может содержать ре-

* Работа выполнена при частичной финансовой поддержке Министерства образования и науки РФ по Соглашению о предоставлении субсидии № 14.578.21.0006 от 05.06.2014, уникальный идентификатор RFMEFI57814X0006

конфигурируемые вычислительные узлы и узлы универсальных микропроцессоров, в роли которых могут выступать универсальные процессоры, графические процессоры или ускорители Intel Xeon Phi [2]. В настоящее время для программирования таких вычислительных систем зачастую используются технологии программирования гетерогенных вычислительных систем: CUDA [3], OpenACC, OpenCL [4] и т.д., в основе которых лежат расширения языков программирования C, C++, FORTRAN, учитывающие архитектуру специализированного микропроцессорного узла. К существенным недостаткам этих технологий программирования относятся плохая переносимость готовых решений между ВС различной архитектуры и конфигурации и плохая масштабируемость программ. Основной причиной указанных недостатков, по нашему мнению, является подход к программированию ВС, при котором осуществляется разбиение задачи на отдельные фрагменты, каждый из которых реализуется на отдельном узле (на отдельном устройстве) гибридной вычислительной системы. Таким образом, выполняется независимое программирование каждого задействованного узла ВС, в результате чего любое изменение конфигурации ВС или изменение исходного кода прикладной программы приводит к необходимости повторного переразбиения задачи на фрагменты и созданию локальных программ для каждого узла ВС.

Можно сформулировать основные проблемы программирования современных ВСГТ, содержащих реконфигурируемые и микропроцессорные вычислительные узлы:

1. ПЛИС и микропроцессоры программируются на разных языках программирования независимо друг от друга.
2. Прикладная программа пишется под текущую конфигурацию ВСГТ, любое изменение структуры системы приводит к изменениям программы.
3. Синхронизация информационных потоков в структуре задачи возлагается на программиста.
4. Портирование прикладной программы на другую систему с похожей конфигурацией приводит к полной переработке программы.
5. Время программирования и отладки прикладной задачи для ВСГТ составляет от 6 до 12 месяцев**.**

Поэтому для программирования ВСГТ необходимы как средства описания различных вариантов организации вычислений в едином для различных архитектур языковом пространстве, так и средства трансляции параллельных прикладных программ, объединенные в технологию ресурсонезависимого программирования ВСГТ, под которой будем понимать совокупность знаний, методов, технологических приемов и средств, которая обеспечивает возможность гибкого изменения и масштабирования программы под новую вычислительную архитектуру или конфигурацию вычислительной системы.

Для обеспечения функционирования унифицированных процессорных и реконфигурируемых вычислительных узлов в едином контуре необходима новая технология ресурсонезависимого программирования ВСГТ [5], базирующаяся на следующих принципах:

- адаптация программы под текущую конфигурацию ВСГТ выполняется в автоматизированном режиме специальной программой - препроцессором на основе методов редукции производительности [6];

- определение эффективных параметров масштабирования и редуцирования производительности должно выполняться без участия пользователя с помощью автоматизированных средств программирования;

- для автоматизированного преобразования под текущую конфигурацию ВСГТ прикладная программа должна быть представлена в канонической форме (единой параллельно-конвейерной форме).

Преобразование программы в единую параллельно-конвейерную форму, обеспечивающую возможность как увеличения параллелизма задачи (индукцию) при увеличении аппаратного ресурса, так и возможность сокращения (редукции) при сокращении вычислительного ресурса, является основой для применения автоматизированных средств. Для реализации технологии ресурсонезависимого программирования ВСГТ необходимо выбрать язык программирования, который позволит описывать различные формы организации вычислений и программировать унифицированные процессорные и реконфигурируемые вычислительные узлы в едином вычислительном контуре.

Специализированные языки высокого уровня для программирования реконфигурируемых вычислительных систем (РВС) обладают привычным для большинства программистов персональных ЭВМ синтаксисом языка С и отличаются между собой семантическими особенностями вызова и использования операторов [7]. Для описания параллельных процессов в РВС в этих языках используется изначально последовательная парадигма языка С, семантика которого ориентирована на взаимодействие последовательных процессов, что и не позволяет в полной мере использовать все возможности РВС при разработке параллельных программ на этих языках. Это приводит к семантическому разрыву между исходным информационным графом задачи, его описанием на языке высокого уровня и созданной транслятором схемотехнической реализацией. Результатом этого разрыва является существенное снижение эффективности параллельной программы - как правило, в 3-5 раз более низкая производительность по сравнению с приложениями, разработанными на языках HDL-группы [8].

Перспективным направлением в области программирования РВС является язык высокого уровня COLAMO[9], разрабатываемый в НИИ МВС ЮФУ. Язык COLAMO предназначен для описания реализации параллельного алгоритма и создания на основе принципов структурно-процедурной организации вычислений специализированной вычислительной структуры в архитектуре РВС, которая выполняет последовательную смену структурно (аппаратно) реализованных фрагментов информационного графа задачи, каждый из которых является вычислительным конвейером потока операндов. Таким образом, приложение (прикладная задача) для РВС состоит из структурной составляющей, представленной в виде аппаратно реализованных фрагментов вычислений, и процедурной составляющей, представляющей собой единую для всех структурных фрагментов управляющую программу последовательной смены вычислительных структур и организации потоков данных.

Для реализации вычислений на универсальных процессорах в языке COLAMO есть средства описания процедурной организации вычислений и возможность быстрого перехода от процедурной реализации вычислений на универсальных процессорах к структурной организации вычислений на реконфигурируемых вычислительных узлах. Конструкция Implicit позволяет неявным образом указать тип организации вычислений (структурный или процедурный) для фрагмента программы. Переопределение способа реализации конструкции Implicit позволяет прикладному программисту без существенного изменения текста параллельной программы перейти от структурной организации вычислений к процедурной и обратно, что дает возможность программисту создавать единую прикладную программу на одном языке для всех узлов ВСГТ. Это позволяет рассматривать язык программирования высокого уровня COLAMO как основу для реализации технологии ресурсонезависимого программирования как реконфигурируемых вычислительных узлов, так и универсальных узлов ВСГТ.

Однако для эффективного программирования ВСГТ языковые средства должны иметь возможность описания фрагментов вычислений, работающих с различными частотой, скважностью и разрядностью обрабатываемых данных для обеспечения масштабирования как фрагментов, так и отдельных устройств не только при увеличении аппаратного ресурса, но и при его сокращении, а также возможность работы с данными переменной разрядности для эффективного использования аппаратного ресурса ВСГТ.

3. Единая параллельно-конвейерная форма программы на языке COLAMO для ВСГТ

Под параллельно-конвейерной формой представления переменных и массивов здесь и далее понимается описание данных на языке программирования высокого уровня COLAMO, обладающее одновременно как параллельным, так и последовательным типами доступа. На языке высокого уровня COLAMO такие типы доступа задаются с помощью ключевых слов Vector (BitVector) и Stream (BitStream)[10]. На рисунке 1 показан пример одновременного использования параллельного и последовательного типов доступа по данным и по разрядам к массивам A, B и C.

```
Const N = 10;
Const M = 100;
Const BV = 32;
Const BS = 1;
Type Type32 [BV : BitVector, BS : BitStream,] of Int;
Var a, b, c : Array Type32 [N: Vector, M: Stream] Mem;
Var i, j, k, t : Number;
Cadr ExpParallelStream;
   For i := 0 to N - 1 do
    For j := 0 to M - 1 do
      For k := 0 to BV - 1 do
        For t := 0 to BS - 1 do
            Begin
              C[i,j][k, t] := A[i,j][k, t] - B[i,j][k, t];
            end;
EndCadr;
```

Рис. 1. Одновременное использование параллельного и последовательного типов доступа к массивам по данным и разрядам

Такая форма представления программы является канонической формой и позволяет автоматизировано менять основные параметры параллельной программы (число одновременно реализуемых подграфов вычислений, разрядность обрабатываемых данных, число операций и др.). Это можно выполнить с помощью программы-препроцессора без участия пользователя для адаптации под текущую конфигурацию ВСГТ.

Приведение исходной программы к канонической форме представления выполняется в два этапа. На первом этапе выполняется преобразование переменных и конструкций программы к параллельно-конвейерной обработке на уровне данных, а на втором этапе – на уровне разрядов. В общем виде метод преобразования программы к канонической форме представления можно представить следующим образом.

1. Все массивы в исходной программе на языке программирования высокого уровня COLAMO преобразуются в ПКФ (при необходимости добавляются параллельный (Vector) или последовательный (Stream) типы доступа).

2. Все переменные параллельной программы на языке программирования высокого уровня COLAMO (кроме счетчиков цикла) преобразуются в формат конструкции union, содержащей механизм как непосредственного обращения к переменной по её типу, так и параллельный (bitvector) и последовательный (bitstream) типы доступа.

3. Все подкадры из параллельной программы на языке COLAMO преобразуются в конструкции Implicit.

4. Все конструкции и операторы программы преобразуются согласно модифицированным параметрам переменных.

5. В сформированной в единой ПКФ форме программы на языке программирования высокого уровня COLAMO задаётся глобальная директива препроцессора редукции производительности по функциональным устройствам, равная 1.

Преобразование переменных к параллельному и последовательному типу доступа (смешанному типу доступа) выполняется согласно следующих правил:

- если для доступа к элементам массива использовался только последовательный тип доступа, то в объявлении массива добавляется параметр Vector единичной размерности, описывающий параллельный тип доступа;
- если для доступа к элементам массива использовался только параллельный тип доступа, то в объявлении массива добавляется параметр Stream единичной размерности, описывающий последовательный тип доступа;
- если для доступа к элементам массива использовался смешанный тип доступа, то преобразования для данного массива не выполняются.

Таким образом, при модификации описания массива выполняется расширение его размерности. Так, при объявлении массива A следующего вида:

Var A : Array Integer [N : Stream] Mem

должно быть выполнено преобразование к объявлению вида

Var A : Array Integer [M : Vector, K : Stream] Mem, где $M * K = N$.

Для обеспечения эквивалентности информационных графов исходной программы и модифицированной программы начальное значение M=1.

На рис. 2 показан пример преобразования исходной программы а) к канонической форме б).

Как видно из текста параллельно-конвейерной программы (рис. 2-б), все одномерные массивы были преобразованы к двумерным массивам, имеющим смешанный тип доступа (M : Vector, K : Stream), добавлен новый оператор цикла с индексной переменной VC_1 и выполнена модификация всех обращений к переменным.

```
Const N = 10;
Var a, b, c, d : Array Integer [N: Vector] Mem;
Var i : Number;
Cadr ExpParallel;
  For i := 0 to N - 1 do
   Begin
     If (A[i] > 5)
       C[i] := A[i] - B[i];
     Else
       C[i] := A[i] + D[i];
   end;
EndCadr;
```

а) Исходная программа

```
Const N = 10;
Const M = 10;
Const K = N/M;
Var a, b, c, d : Array Integer [M : Vector, K :
Stream] Mem;
Var i, vc_1 : Number;
Cadr ExpParallelConvData;
  For vc_1 := 0 to K - 1 do
   Begin
    For i := 0 to M - 1 do
     Begin
       If (A[i, vc_1] > 5)
         C[i, vc_1] := A[i, vc_1] - B[i, vc_1];
       Else
         C[i, vc_1] := A[i, vc_1] + D[i, vc_1];
     end;
   end;
EndCadr;
```

б) Преобразованная программа

Рис. 2. Преобразование программы к канонической форме на уровне данных

Для эффективной адаптации программы для ВСГТ такие же преобразования должны быть выполнены и для разрядов.

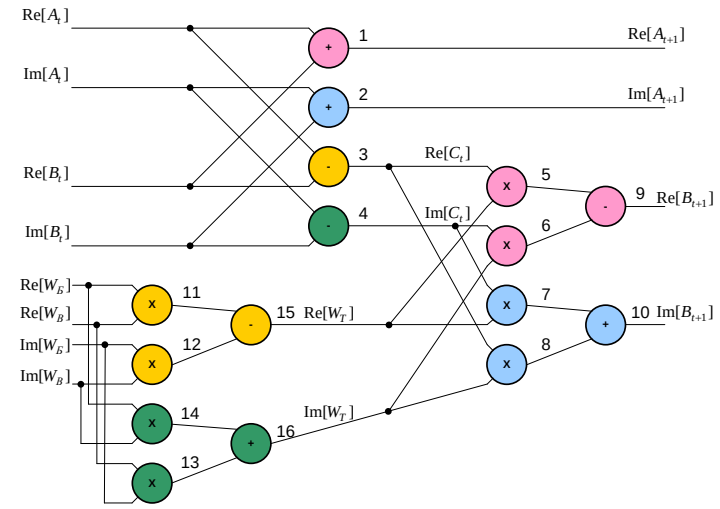
4. Масштабирование вычислений в ВСГТ на основе редукции производительности

Основой для простого масштабирования и адаптации прикладной программы как для случая увеличения, так и для случая сокращения доступного аппаратного ресурса, является редукция производительности[6] прикладной программы, под которой понимается пропорциональное сокращение производительности во всех без исключения фрагментах информационного графа задачи с возможным сокращением аппаратных затрат на реализацию вычислительной структуры. Редуцирование производительности параллельных программ позволяет изменять ключевые параметры параллельных программ (число задействованных вычислительных устройств, число каналов памяти, разрядность операндов, частота и др.), поскольку структурная реализация задачи может привести к нехватке доступного аппаратного ресурса, что особенно актуально при переносе задачи на ВСГТ различных архитектур и конфигураций. В отличии от традиционных технологий и методов программирования многопроцессорных вычислительных систем (MPI, CUDA, OpenAcc и др.), которые предполагают распараллеливание базового информационного подграфа прикладной задачи в зависимости от конфигурации доступного вы-

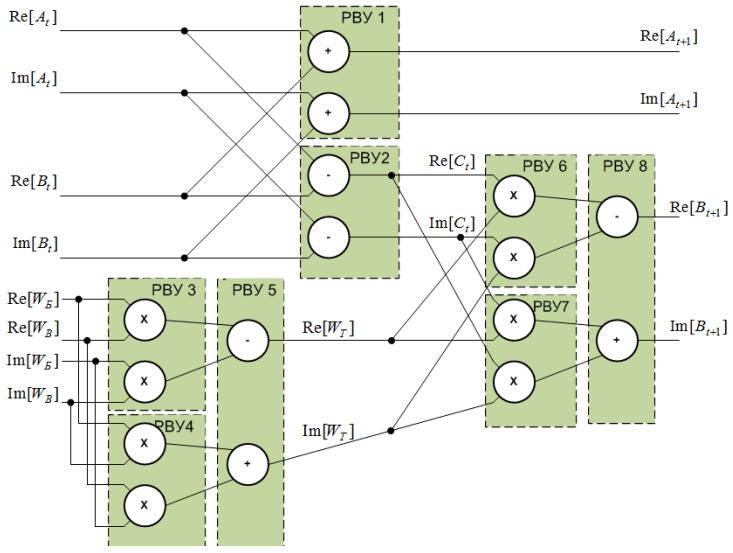
числительного ресурса, для применения редукции производительности информационный граф задачи описывается в исходной параллельной форме с максимальным присущим задаче параллелизмом. В зависимости от числа доступных вычислительных узлов ВСГТ исходный информационный граф сокращается (редуцируется) с помощью специальных *редукционных* преобразований, которые сбалансированно сокращают производительность всех фрагментов информационного графа и, в ряде случаев, сокращают занимаемый задачей аппаратный ресурс ВСГТ. Можно выделить следующие виды редукции:

- редукция производительности по вычислительным устройствам;
- редукция производительности по каналам памяти;
- редукция производительности по разрядности;
- редукция производительности по частоте.

Редукция производительности по вычислительным устройствам основана на сокращении одновременно выполняемых устройств, реализующих вычислительные операции. Иллюстрация принципов функционирования редукции производительности по вычислительным устройствам на примере операции быстрого преобразования Фурье (БПФ) представлена на рис. 3 и 4.



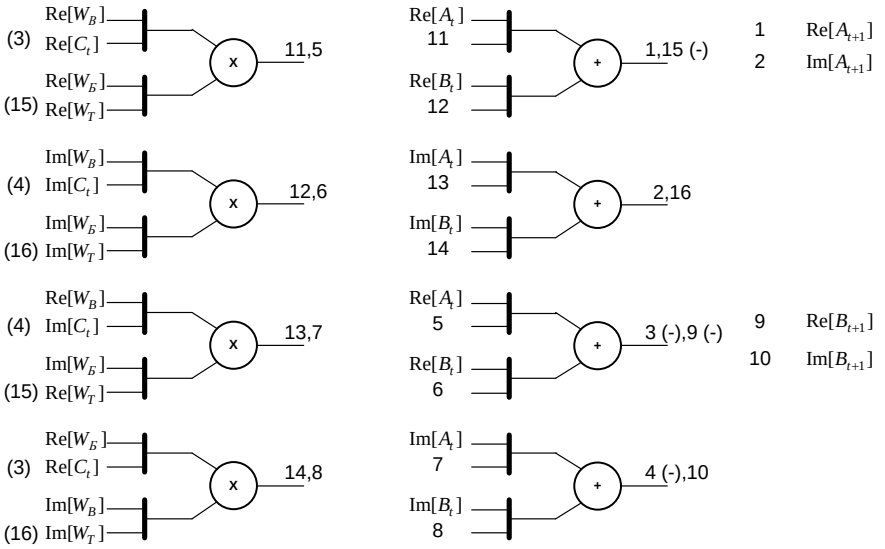
а)



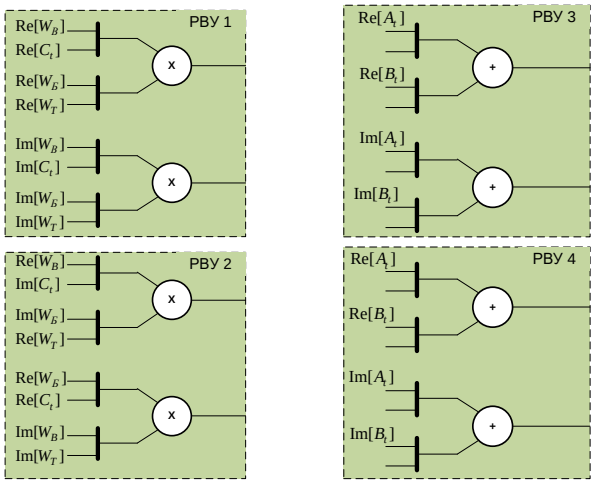
б)

Рис. 3. Принципы функционирования редукции производительности на примере операции БПФ (а - исходный информационный граф операции БПФ, б - структурная реализация операции БПФ на РВС с условным заполнением ПЛИС)

На рис. 3а представлен исходный информационный граф операции быстрого преобразования Фурье, содержащий 16 вычислительных устройств, а на рис. 3б - структурная реализация этой операции на РВС с условным заполнением ПЛИС (в намеренном допущении, что два вычислительных устройства занимают ресурс одного условного кристалла ПЛИС). При выполнении редукции производительности по функциональным устройствам осуществляется пропорциональное сокращение числа одновременно работающих устройств, что приводит как к сокращению производительности, так и к сокращению занимаемого аппаратного ресурса. Степень редукции производительности задается коэффициентом редукции, который определяет во сколько раз сокращается производительность фрагмента, а для рассматриваемого примера также определяет во сколько раз сокращается число функциональных устройств.



а)



б)

Рис. 4. Результат редукции производительности по функциональным устройствам со степенью 2 для базовой операции БПФ (а – редуцированный в 2 раза информационный граф операции БПФ, б - структурная реализация операции БПФ на РВС с условным заполнением ПЛИС)

Результат редукции производительности по функциональным устройствам со степенью 2 для базовой операции быстрого преобразования Фурье представлен на рис. 4. В представленном на рис. 4 примере видно, что число функциональных устройств сократилось вдвое (до 8), и результат операции получается не за один такт работы (для структурной реализации, см. рис. 3), а за два такта работы.

На языке высокого уровня один и тот же информационный граф можно описать различны-

ми способами, поэтому одно и то же редукционное преобразование может быть выполнено различными способами, которые будем называть *механизмами* редукции. Так, редукция производительности по вычислительным устройствам для рассмотренного примера может быть реализована в виде:

1) однокадровой параллельной программы с использованием мультиплексоров и скважности, равной степени выполняемой редукции;

2) многокадровой программы с использованием конструкции Let;

3) мультикадровой программы с использованием конструкции MultiCadr.

Выбор используемого механизма редукции осуществляет программист при описании редукции.

Одним из важнейших типов редукции, обеспечивающих возможность ресурсонезависимого программирования ВСГТ, является редукция производительности по каналам памяти. Редукция производительности по каналам памяти представляет собой согласованное сокращение числа одновременно используемых каналов памяти для фрагмента информационного графа прикладной задачи. Иллюстрация принципов функционирования редукции производительности по каналам памяти представлена на рис. 5 и 6.

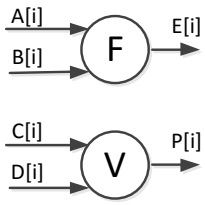


Рис. 5. Исходный информационный граф редуцируемого фрагмента

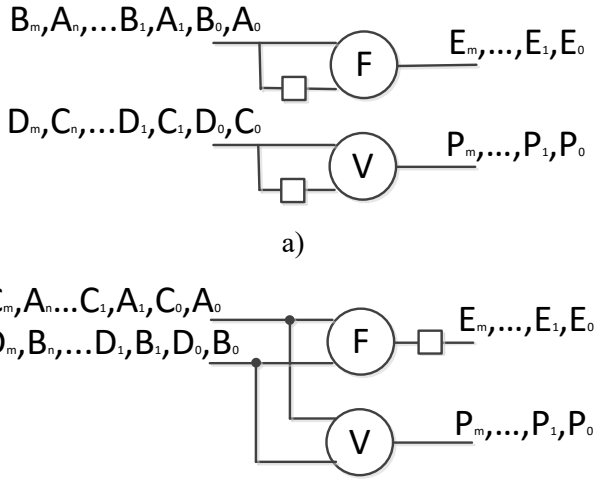


Рис. 6. Результат редукции производительности по каналам памяти со степенью 2 (а - редукция производительности по каналам с размещением данных для каждого вычислительного устройства в одном канале, б - редукция производительности по каналам с размещением данных в разных каналах памяти для каждого вычислительного устройства)

На рис. 5 представлен фрагмент исходного информационного графа, содержащий 4 входных канала, а на рис. 6 – результат выполнения редукции производительности по каналам памяти со степенью 2. При выполнении редукции производительности по каналам памяти число одновременно работающих устройств не изменяется, но время обработки потоков данных возрастает в 2 раза, что в 2 раза сокращает производительность фрагмента.

Редукция производительности по разрядности направлена не на сокращение устройств в вычислительной структуре, а на сокращение разрядности обрабатываемых данных за счет использования устройств меньшей разрядности, что приводит к увеличению времени обработки и сокращению аппаратных затрат на реализацию вычислительной структуры. При выполнении

редукции производительности по разрядности управление информационными потоками данных осуществляется мультиплексором, а сами данные подаются в вычислительную структуру со скважностью, значение которой равно степени выполняемой редукции.

Редукция производительности по частоте предназначена для увеличения времени обработки потока данных пропорционально степени выполняемой редукции за счет кратного сокращения частоты работы фрагмента, при этом скважность подачи данных в вычислительную структуру остается неизменной. Редукция производительности является служебным редукционным преобразованием, которое используется не самостоятельно, а применяется для согласования скоростей обработки между редуцированными и нередуцированными фрагментами информационного графа в структуре прикладной задачи.

Для практического использования рассмотренных редукционных преобразований программист должен разметить директивами препроцессора те фрагменты параллельной программы на языке высокого уровня COLAMO, которые могут быть редуцированы. Формат описания директивы редукции может выглядеть следующим образом:

#Reduction of <*вид редукции*> <*степень редукции*>;

Блок операторов

EndReduction;

При трансляции параллельной программы препроцессор в соответствии с расставленными программистом директивами редукции в автоматическом режиме преобразует информационный граф прикладной программы под доступную конфигурацию ВСГТ, что позволит адаптировать прикладную программу к текущей конфигурации ВСГТ без существенной модернизации исходного текста программы.

Подводя итог рассмотренным преобразованиям, функционирование технологии ресурсонезависимого программирования вычислительных систем гибридного типа, содержащих реконфигурируемые и микропроцессорные вычислительные узлы, можно представить следующими действиями. Модуль анализа исходной параллельной программы препроцессора языка COLAMO преобразует параллельную программу в каноническую форму, после чего подсчитывает необходимый для ее реализации аппаратный ресурс и сопоставляет его с текущей конфигурацией ВСГТ для определения максимальной степени и типов необходимых преобразований. Если текущего аппаратного ресурса ВСГТ достаточно для выполнения программы, то синтезируются загрузочные модули для задействованных узлов ВСГТ, в противном случае выполняются специальные редукционные преобразования, сбалансированно сокращающие производительность программы и задействованный аппаратный ресурс ВСГТ. Редукция производительности выполняется в следующем порядке: редукция по одновременно выполняемым подграфам программы, редукция по разрядности, редукция по командам (устройствам), редукция по скважности (частоте). Сокращение занимаемого аппаратного ресурса для каждого вида редукции с учетом ее теоретически допустимой степени выполняет модуль анализа препроцессора, который определяет наиболее рациональный вариант использования редукции для обеспечения максимально возможной производительности программы для текущей конфигурации ВСГТ. Полученный в автоматизированном режиме после препроцессора текст редуцированной параллельной программы передается транслятору языка программирования COLAMO, который создает развернутый информационный граф прикладной задачи. Информационный граф прикладной задачи, содержащий фрагменты, реализуемые структурно на реконфигурируемых вычислительных узлах и процедурно на микропроцессорных вычислительных узлах, передается программе-синтезатору для автоматического распределения фрагментов задачи по доступным в текущей конфигурации ВСГТ реконфигурируемым и микропроцессорным вычислительным узлам. Согласование потоков данных между различными по типам организации вычислений узлам ВСГТ осуществляется на основе файла описания конфигурации ВСГТ, содержащего данные о типах синхронизируемых вычислительных узлов, разрядности шины данных, частоте их работы, скважности подачи данных и др. После установки необходимых интерфейсов и элементов синхронизации программа-синтезатор создает загрузочные конфигурационные файлы *.bit для реконфигурируемых вычислительных узлов и загрузочные файлы *.exe для микропроцессорных узлов ВСГТ и единую для всех вычислительных модулей ВСГТ управляющую вычислительным процессом программу.

5. Заключение

Для эффективного программирования вычислительных систем гибридного типа в рамках разрабатываемой технологии ресурсонезависимого программирования предложен язык программирования высокого уровня COLAMO, позволяющий описывать в едином вычислительном контуре различные формы организации параллельных вычислений. Предложенная единая параллельно-конвейерная форма прикладной программы в совокупности с разработанными методами редукции производительности позволяет автоматизированно адаптировать прикладную программу под изменившуюся архитектуру или конфигурацию ВСГТ. Предложенная технология позволяет рационально использовать ресурсы узлов с разной архитектурой при программировании ВСГТ и обеспечивает пользователя набором необходимых средств для быстрой разработки эффективных ресурсонезависимых масштабируемых параллельных программ в едином языковом пространстве, что снижает сложность программирования ВСГТ и повышает скорость разработки параллельных прикладных программ.

Литература

1. I. A. Kalyaev, I. I. Levin, E. A. Semernikov, V. I. Shmoilov, Reconfigurable multipipeline computing structures. New York, Nova Science Publishers, 2012.
2. Dong X, Chai J, Yang J, Wen M, Wu N, Cai X, Zhang C, Chen Z. Utilizing multiple xeon Phi co-processors on one compute node // 14th International Conference on Algorithms and Architectures for Parallel Processing, ICA3PP 2014; Dalian; China; 24 August 2014 through 27 August 2014; Code 107001. Volume 8631 LNCS, Issue PART 2, 2014, Pages 68-81.
3. Liang T.-Y., Li H.-F., Lin Y.-J., Chen B.-S. A Distributed PTX Virtual Machine on Hybrid CPU/GPU Clusters // Journal of Systems Architecture. – Volume 62, 1 January 2016, Pages 63-77.
4. Li H.-F., Liang T.-Y., Lin Y.-J. An OpenMP programming toolkit for hybrid CPU/GPU clusters based on software unified memory // Journal of Information Science and Engineering. – Volume 32, Issue 3, May 2016, Pages 517-539.
5. Дордопуло А.И., Гудков В.А., Гуленок А.А., Левин И.И. Ресурсонезависимое программирование вычислительных систем гибридного тип. Materiały XI Międzynarodowej Naukowi-praktycznej Konferencji «Aktualne Problemy nowoczesnych nauk - 2015» 07-15 czerwca 2015 roku Volume 13, Matematyka Fizyka Przemyśl Nauka i studia, 2015. – Pp. 23-26.
6. Aleksey Dordopulo, Ilya Levin, Igor Kalyaev, Vyacheslav Gudkov, Andrey Gulenok Programming of hybrid computer systems based on the performance reduction method. Parallel Computing Technologies (PACT 2016), Proceedings of the 10th Annual International Scientific Conference on Parallel Computing Technologies, Arkhangelsk, Russia, 2016. – Pp. 131-140.
7. El-Araby E., Taher M., Abouellail M., El-Ghazawi T., Newby G.B. Comparative analysis of high level programming for reconfigurable computers: Methodology and empirical study // 2007 3rd Southern Conference on Programmable Logic, SPL'07; Mar del Plata; Argentina; 26 February 2007 through 28 February 2007; Category number07EX1511; Code 70259. 2007, Article number 4234328, Pages 99-106.
8. Xu J, Subramanian N, Alessio A, Hauck S. Impulse C vs. VHDL for accelerating tomographic reconstruction // 18th IEEE International Symposium on Field-Programmable Custom Computing Machines, FCCM 2010; Charlotte, NC; United States; 2 May 2010 through 4 May 2010; Category numberP4056; Code 80904. 2010, Article number 5474054, Pages 171-174.
9. Alexey Dordopulo, Igor Kalyaev, Ilya Levin, Liubov Slasten. High-performance reconfigurable computer systems // Lecture Notes in Computer Science, Volume 6873, Chapter Parallel Computing Technologies, pp 272-283.
10. Семерникова, Е.Е. Организация битовой обработки данных для реконфигурируемых вычислительных систем на языке программирования высокого уровня [Текст] / Е.Е. Семерникова, И.И. Левин, В.А. Гудков // Вестник компьютерных и информационных технологий. – М.: Машиностроение, 2015. - №5. – С. 3-9.

Resource independent programming of hybrid computer systems in the programming language COLAMO*

A.I. Dordopulo[1], I.I. Levin[2], I.A. Kalyaev[1], V.A. Gudkov[2], A.A. Gulenok[1]

[1] Scientific Research Institute of Multiprocessor Computer Systems at Southern Federal University, Taganrog, Russia
[2] Scientific Research Centre of Supercomputers and Neurocomputers, Taganrog, Russia

The paper covers programming methods for hybrid computer systems that contain reconfigurable and microprocessor computational nodes. The technology of resource-independent programming allows description various types of parallel calculations such as structural, structural-procedural, multi-procedural and procedural forms of organization of calculations in a unified parallel-pipeline form for high-level programming language COLAMO. The form of organization of calculations is transformed automatically by COLAMO language preprocessor with the help of the methods of performance reduction.

Keywords: performance reduction, high-level programming language, programming of hybrid computer systems, technology of resource-independent programming.

References

1. I. A. Kalyaev, I. I. Levin, E. A. Semernikov, V. I. Shmoilov, Reconfigurable multipipeline computing structures. New York, Nova Science Publishers, 2012.
2. Dong X, Chai J, Yang J, Wen M, Wu N, Cai X, Zhang C, Chen Z. Utilizing multiple xeon Phi co-processors on one compute node // 14th International Conference on Algorithms and Architectures for Parallel Processing, ICA3PP 2014; Dalian; China; 24 August 2014 through 27 August 2014; Code 107001. Volume 8631 LNCS, Issue PART 2, 2014, Pages 68-81.
3. Liang T.-Y., Li H.-F., Lin Y.-J., Chen B.-S. A Distributed PTX Virtual Machine on Hybrid CPU/GPU Clusters // Journal of Systems Architecture. – Volume 62, 1 January 2016, Pages 63-77.
4. Li H.-F., Liang T.-Y., Lin Y.-J. An OpenMP programming toolkit for hybrid CPU/GPU clusters based on software unified memory // Journal of Information Science and Engineering. – Volume 32, Issue 3, May 2016, Pages 517-539.
5. Dordopolo A.I., Gudkov V.A., Gulenok A.A., Levin I.I. Resourcesavesize programming computing systems hybrid type // Materiały XI Międzynarodowej Naukowi-praktycznej Konferencji «Aktualne Problemy nowoczesnych nauk - 2015» 07-15 czerwca 2015 roku Volume 13, Matematyka Fizyka Przemyśl Nauka i studia, 2015. – Pp. 23-26.
6. Aleksey Dordopulo, Ilya Levin, Igor Kalyaev, Vyacheslav Gudkov, Andrey Gulenok Programming of hybrid computer systems based on the performance reduction method. Parallel Computing Technologies (PACT 2016), Proceedings of the 10th Annual International Scientific Conference on Parallel Computing Technologies, Arkhangelsk, Russia, 2016. – Pp. 131-140.
7. El-Araby E., Taher M., Abouellail M., El-Ghazawi T., Newby G.B. Comparative analysis of high level programming for reconfigurable computers: Methodology and empirical study // 2007 3rd Southern Conference on Programmable Logic, SPL'07; Mar del Plata; Argentina; 26 February 2007 through 28 February 2007; Category number07EX1511; Code 70259. 2007, Article number 4234328, Pages 99-106.
8. Xu J, Subramanian N, Alessio A, Hauck S. Impulse C vs. VHDL for accelerating tomographic reconstruction // 18th IEEE International Symposium on Field-Programmable Custom Computing Machines, FCCM 2010; Charlotte, NC; United States; 2 May 2010 through 4 May 2010; Category numberP4056; Code 80904. 2010, Article number 5474054, Pages 171-174.

* The project has been partly funded by the Ministry of Education and Science of the Russian Federation. Grant agreement № 14.578.21.0006, unique identifier RFMEFI57814X0006

9. Alexey Dordopulo, Igor Kalyaev, Ilya Levin, Liubov Slasten. High-performance reconfigurable computer systems // Lecture Notes in Computer Science, Volume 6873, Chapter Parallel Computing Technologies, pp 272-283.
10. Semernikova. E.E. Organizatsiya bitovoy obrabotki dannykh dlya rekonfiguriruyemykh vychislitelnykh sistem na yazyke programmirovaniya vysokogo urovnya [Tekst] / E.E. Semernikova. I.I. Levin. V.A. Gudkov // Vestnik kompyuternykh i informatsionnykh tekhnologiy. – M.: Mashinostroyeniye. 2015. - №5. – S. 3-9.