15. *Yaglom I.M.* Geometricheskie preobrazovaniya. Ch. 1. Dvizheniya i preobrazovaniya podobiya [The geometric transformation. Part 1. Motions and similarity transformations]. Moscow: Gosudarstvennoe izdatel'stvo tekhniko-teoreticheskoy literatury, 1956, 280 p.
16. *Kuznetsov E.B.* Ob odnom podkhode k integrirovaniyu kinematicheskikh uravneniy Eylera [About one approach to integrating the kinematic Euler equations], *Vychislitel'naya matematika i matematicheskaya fizika* [Computational mathematics and mathematical physics], 1998, Vol. 38, No. 11, pp. 1806-1813.
17. *Konishchev D.* Chto takoe vyravnivanie i kak ono vliyaet na rabotu vashikh program [What is alignment and how it affects your programs]. Available at: http://webcache.googleusercontent.com/search?q=cache:rb4kHpT7ansJ:konishchevdmitry.blogspot.com/2010/01/blog-post.html+&cd=1&hl=en&ct=clnk&gl=ru. (accessed 12 August 2016).
18. *Ermolitskiy A.E.* Metody povysheniya effektivnosti vektorizatsii v optimiziruyushchem kompilyatore [Methods of increasing the efficiency of vectorization in an optimizing compiler], *Voprosy radioelektroniki. Ser. EVT* [Questions of radio electronics. Series of EVT], 2010, Issue 3, pp. 41-50.
19. *Muller J.-M., Brisebarre N*. The Fused Multiply-Add Instructions, *Handbook of Floating-point Arithmetic*, 2009, pp. 151-179.
20. *Zumbusch G.* Vectorized Higher Order Finite Difference Kernels, State-*of-the-Art in Scientific and Parallel Computing (PARA)*, 2012, pp. 343-357.

Статью рекомендовал к опубликованию д.т.н., профессор И.И. Левин.

Гетманский Виктор Викторович – Волгоградский государственный технический университет; e-mail: victor.getmanski@gmail.com; 400005, г. Волгоград, пр. Ленина, 28; тел.: +79275065804; кафедра ЭВМ и систем; с.н.с.; к.т.н.

Мовчан Евгения Олеговна – e-mail: verborum123@mail.ru; тел.: +79197980049; бакалавр; студент.

Андреев Андрей Евгеньевич – e-mail: andan2005@yandex.ru; тел.: +79023628177; кафедра ЭВМ и систем; и.о. зав. кафедрой; к.т.н.

Getmanskiy Victor Victorovich – Volgograd State Technical University; e-mail: victor.getmanski@gmail.com; 28, Lenin av., Volgograd, 400005, Russia; phone: +79275065804; the department of computers and systems; senior researcher; cand. of eng. sc.

Movchan Evgenija Olegovna – e-mail: verborum123@mail.ru; phone: +79197980049; bachelor; student.

Andreev Andrey Evgenevich – e-mail: andan2005@yandex.ru; phone: +79023628177; the department of Computers and systems; head of department; cand. of eng. sc.

УДК 004.4 DOI 10.18522/2311-3103-2016-11-3954

А.И. Дордопуло, И.И. Левин, И.А. Каляев, В.А. Гудков, А.А. Гуленок

ПРОГРАММИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ ГИБРИДНОГО ТИПА НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ COLAMO*

Рассматриваются методы программирования вычислительных систем гибридного типа, содержащих реконфигурируемые и микропроцессорные вычислительные узлы. В качестве основы технологии программирования вычислительных систем гибридного типа предлагается язык программирования высокого уровня COLAMO с расширениями, с помощью которых можно описывать различные виды параллельных вычислений – структурную,

* 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.

структурно-процедурную, мультипроцедурную и процедурную формы организации вычислений в единой параллельно-конвейерной (канонической) форме. Предложенная параллельно-конвейерная форма позволяет изменять формы организации вычислений автоматизированно препроцессором языка COLAMO с учетом текущей конфигурации вычислительной системы гибридного типа. На основе канонической формы и возможностей описания различных форм организации вычислений на языке программирования высокого уровня COLAMO предложена технология ресурсонезависимого программирования, которая позволяет адаптировать программу под изменившиеся архитектуру или конфигурацию вычислительной системы гибридного типа в автоматическом режиме без корректировки кода программистом. Для этого исходная параллельная программа на языке программирования COLAMO препроцессором преобразуется в каноническую форму (все массивы и переменные программы должны поддерживать как параллельный, так и последовательный типы доступа как к элементам, так и к битам, все фрагменты вычислений приводятся к неявному описанию реализации конструкцией Implicit*), после чего препроцессор проводит оценку доступного вычислительного ресурса, определяет эффективные параметры реализации программы на доступном ресурсе и, при необходимости, выполняет редукцию производительности программы для адаптации под текущую конфигурацию вычислительной системы гибридного типа. Редукция производительности программы – комплекс методов, сбалансированно сокращающих производительность прикладной задачи, что в ряде случаев приводит к сокращению занимаемого задачей аппаратного ресурса и позволяет изменением организации вычислений задействовать свободные узлы вычислительной системы гибридного типа. Технология позволяет осуществлять масштабирование в обе стороны как в случае увеличения доступного вычислительного ресурса (индукция), так и в случае сокращения доступного вычислительного ресурса (редукция), что обеспечивает ресурсонезависимость программирования при разработке программы – программист не привязывается к доступному аппаратному ресурсу вычислительной системы.*

Редукция производительности; язык программирования высокого уровня; программирование вычислительных систем гибридного типа; технология ресурсонезависимого программирования.

A.I. Dordopulo, I.I. Levin, I.A. Kalyaev, V.A. Gudkov, A.A. Gulenok

PROGRAMMING OF HYBRID COMPUTER SYSTEMS IN THE PROGRAMMING LANGUAGE COLAMO

The paper covers programming methods for hybrid computer systems which contain reconfigurable and microprocessor computational nodes. The base of the programming technology for hybrid computer systems is the high-level programming language COLAMO with extensions, which allow descriptions of 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. The suggested parallel-pipeline form allows modifications of forms of the calculations organization. Such modifications are performed automatically by the COLAMO language preprocessor, which takes into account current configuration of the hybrid computer system. On the base of the canonical form and possibilities of description of various forms of the calculations organization in the high-level programming language COLAMO we suggest a technology of resource independent programming. Owing to the suggested technology, the program can be automatically adapted to the changed architecture or configuration of the hybrid computer system without any modifications of the source code made by the developer. Especially for this the source parallel program, developed in the programming language COLAMO, is transformed by the preprocessor into the canonical form (all arrays and variables of the program must provide both parallel and sequential access to both items and bits, all fragments of calculations are described as implicitly parallel by means of the structure Implicit*). Then the pre-processor estimates the available computational resource, detects effective parameters of implementation of the program on the available resource and, if necessary, reduces the program performance to adapt it to the*

current configuration of the hybrid computer system. The performance reduction is a complex of methods, which in a balanced way reduce the performance of the application. In several cases it leads to reduction of hardware resource taken by the task, and besides, owing to change of organization of calculations, it becomes possible to occupy free nodes of the hybrid computer system. The technology provides two-way scaling: for increasing of the available computational resource (induction), and for reducing the available computational resource (reduction), which provides resource independence of programming during implementation of the program, i.e. the developer is not "bound" to the available hardware resource of the computer system.

Performance reduction; high-level programming language; programming of hybrid computer systems; technology of resource-independent programming.

Введение. Большинство реальных практических задач, решаемых на современных высокопроизводительных вычислительных системах, требует совмещения в едином вычислительном контуре как последовательных, так и параллельных вычислительных фрагментов [1] для эффективной реализации структурных и процедурных 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 [2–5] в одной вычислительной системе теоретически позволяет повысить реальную производительность вычислительной системы за счет 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.

Программирование вычислительных систем гибридного типа. Вычислительная система гибридного типа (ВСГТ) содержит различные по архитектуре и типу организации вычислений узлы: реконфигурируемые 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, OpenACC, OpenCL 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 вычислительными системами (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 (или отдельном устройстве) ВСГТ. Таким образом, каждый задействованный узел ВС программируется независимо, поэтому любое изменение конфигурации ВС или изменение исходного кода прикладной программы приводит к необходимости повторного переразбиения задачи на фрагменты и разработке

новых локальных программ для каждого узла ВС. Можно сформулировать основные проблемы программирования современных ВСГТ, содержащих реконфигурируемые и микропроцессорные вычислительные узлы:

- ПЛИС и микропроцессоры программируются на разных языках программирования независимо друг от друга [6–8];
- прикладная программа пишется под текущую конфигурацию ВСГТ, любое изменение структуры системы приводит к изменениям программы [9];
- синхронизация информационных потоков в структуре задачи возлагается на программиста;
- портирование прикладной программы на другую систему с похожей конфигурацией приводит к полной переработке программы;
- 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 [10], 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, 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 (индукция) при увеличении аппаратного ресурса, так и возможность сокращения (редукция) при сокращении вычислительного ресурса, является основой для применения автоматизированных средств. Для реализации технологии ресурсонезависимого программирования ВСГТ необходимо использовать 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, 8]. 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 [11], 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 [12] как реконфигурируемых вычислительных узлов [13], так и универсальных узлов [14] 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.

Единая параллельно-конвейерная форма программы на языке COLAMO для ВСГТ. Под параллельно-конвейерной формой [12] представления переменных и массивов здесь и далее понимается описание данных и программных конструкций на языке программирования высокого уровня COLAMO, обладающее одновременно как параллельным, так и последовательным типами доступа. На языке высокого уровня COLAMO параллельный и последовательный типы доступа на уровне данных задаются с помощью ключевых слов Vector и Stream, а на уровне разрядов – ключевыми словами BitVector и BitStream [10]. На рис. 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: Integer [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] and 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 автоматизированно менять основные параметры параллельной программы (число одновременно реализуемых подграфов вычислений, разрядность обрабатываемых данных, число операций и др.). Это можно выполнить для адаптации под текущую конфигурацию ВСГТ с помощью программы-препроцессора без участия пользователя.

Приведение исходной программы к канонической форме представления выполняется в два этапа. На первом этапе выполняется преобразование переменных и конструкций программы к форме параллельно-конвейерной обработки 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 программы на языке программирования высокого уровня COLAMO задаётся глобальная директива препроцессора редукции производительности по функциональным устройствам, равная 1.

Преобразование переменных к параллельному и последовательному типам доступа (смешанному типу доступа) выполняется согласно следующим правилам:

- 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).

```
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

Как видно из текста параллельно-конвейерной программы в ПКФ (рис. 2,б), все одномерные массивы с параллельным типом доступа (N : Vector) были преобразованы к двумерным массивам, имеющим смешанный тип доступа (M : Vector, K : Stream), а также выполнена модификации всех обращений к данным массивам с использованием нового оператора цикла с индексной переменной VC_1.

Для эффективной адаптации программы для ВСГТ такие же преобразования должны быть выполнены и для разрядов путем перехода от статических типов данных (Integer, Real, Int64 и т.д.) к пользовательским типам данных с указанием параллельного (BitVector) и последовательного (BitStream) типов доступа к разрядам.

Масштабирование вычислений в ВСГТ на основе редукции производительности. Основой для простого масштабирования [15–19] и адаптации [20] прикладной программы как для случая увеличения, так и для случая сокращения доступного аппаратного ресурса, является редукция производительности [10] 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 вычислительного 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 производительности: по вычислительным устройствам, по каналам памяти, по разрядности, по частоте [6].

Редукция производительности по вычислительным устройствам основана на сокращении числа одновременно работающих устройств, реализующих вычислительные операции. Рассмотрим принципы функционирования редукции производительности по вычислительным устройствам на примере базовой операции быстрого преобразования Фурье (БО БПФ), информационный граф базового подграфа которой представлен на рис. 3.

Если проанализировать графическую структуру БО БПФ [19], то можно увидеть, что среди 16 операционных вершин используются три различные арифметические операции: сложение, вычитание и умножение. Поэтому для реализации БО БПФ необходимо как минимум по одному вычислительному устройству, реализующему указанные операции. Можно определить редуцированный базовый подграф, содержащий операции указанных типов и меньшее (по сравнению с рис. 3) число операционных вершин. Сокращение числа операционных вершин (вычислительных устройств при реализации БО БПФ) приведет к увеличению времени решения задачи, но при этом сократит занимаемый аппаратный ресурс.

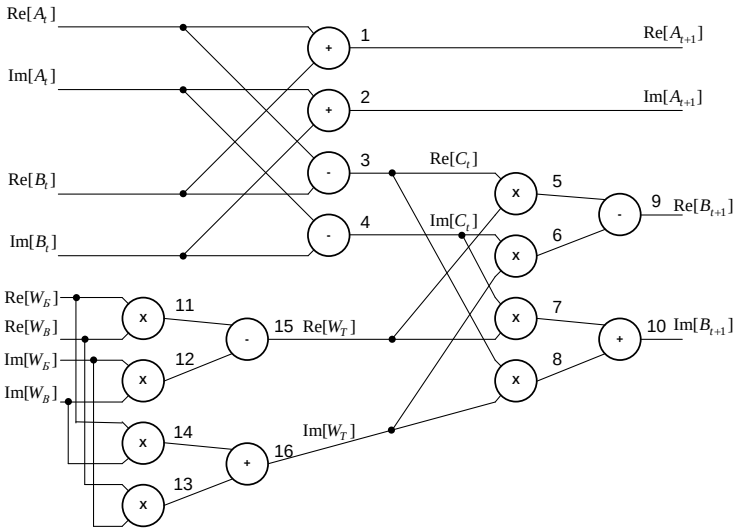
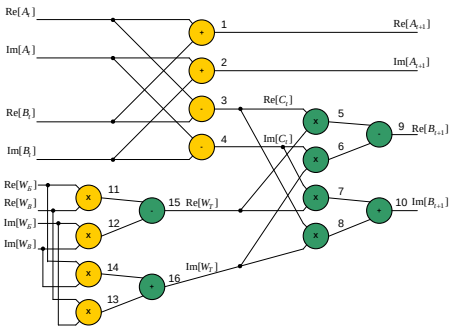
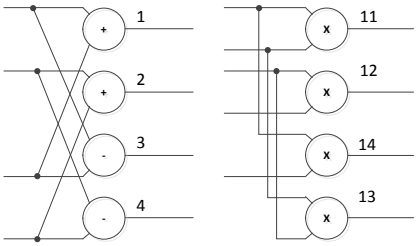


Рис. 3. Исходный информационный граф базовой операции БПФ

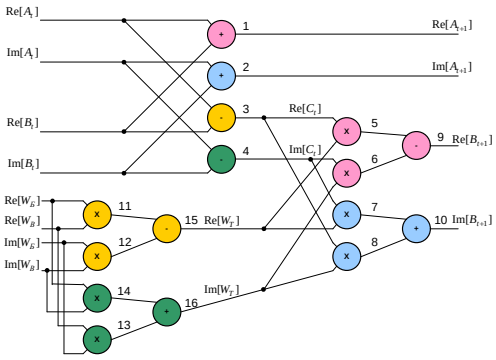
Примеры базовых подграфов с сокращенным числом операционных вершин в два раза (а и б) и четыре раза (в и г) представлены на рис. 4.



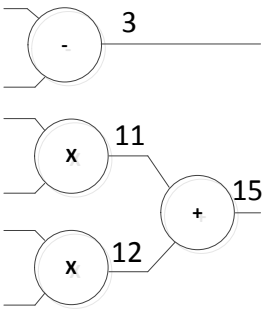
а) Структура из 2-х базовых подграфов



б) Базовый подграф из 8 элементов



в) Структура из 4-х базовых подграфов



г) Базовый подграф из 4-х элементов

Рис. 4. Базовые подграфы БО БПФ

На рис. 4 светлые операционные вершины (рис. 4,б,г) описывают базовый подграф, а вершины других цветов (рис. 4,а,в) – вершины, которые будут покрыты вершинами базового подграфа и удалены в результате выполнения редукции по функциональным устройствам.

Из рис. 4,а,б видно, что количество функциональных устройств в информационном графе может быть сокращено в два раза при использовании в качестве базового подграфа графа (рис. 4,б) или в 4 раза при использовании базового подграфа (рис. 4,г).

Информационный граф при редукции производительности по функциональным устройствам со степенью 2 для базовой операции быстрого преобразования Фурье представлен на рис. 5. В представленном на рис. 5 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. Для управления информационными потоками данных используются коммутаторы, которые каждый такт выполняют переключение обрабатываемых потоков данных.

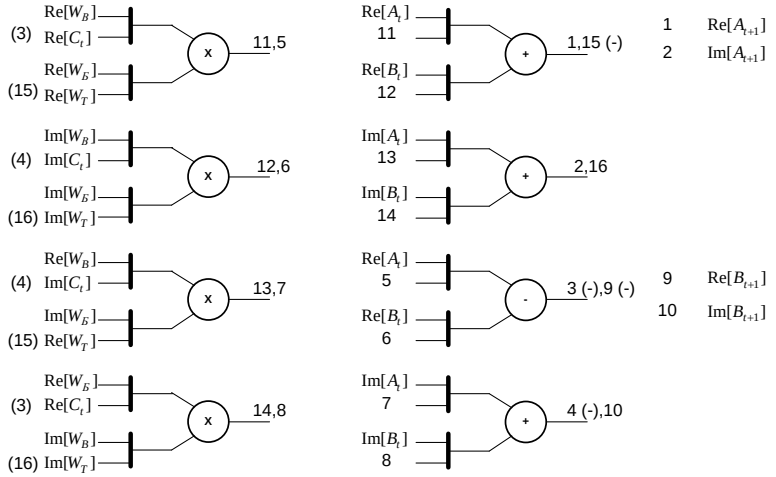
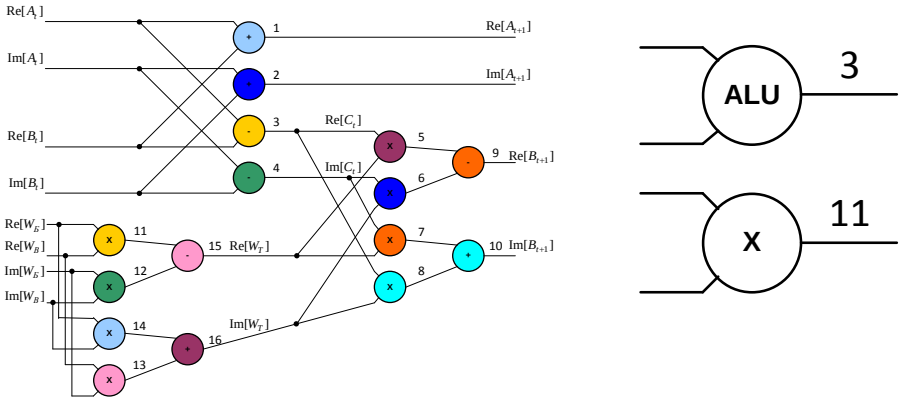


Рис. 5. Результат редукции производительности по функциональным устройствам со степенью 2 для базовой операции БПФ

Таким образом, в зависимости от выбранной структуры базового подграфа можно выполнить редуцирование исходного информационного графа БО БПФ по функциональным устройствам со степенью 2 и 4, используя один и тот же набор арифметических операций (умножение, сложение и вычитание).

Современные возможности схемотехнической реализации позволяют объединить реализацию операций сложения и вычитания в один общий блок – арифметико-логическое устройство (АЛУ) [1], при этом тип выполняемой операции определяется внешним сигналом, подаваемым на вход АЛУ: 0 – выполняется операция вычитания, 1 – операция сложения.

Использование АЛУ позволяет сократить количество типовых операций в минимальном базовом подграфе до двух: умножение и сложение\вычитание. На рис. 6,а представлена структура БО БПФ, состоящая из 8 базовых подграфов, приведенных на рис. 6,б.



а) Структура из 8 базовых подграфов

б) Базовый подграф из 2-х элементов

Рис. 6. Базовый подграфы БО БПФ с использованием АЛУ

Рассмотренный на рис. 6 информационный граф демонстрирует пример редуцирования информационного графа БО БПФ по функциональным устройствам со степенью редукции, равной 8. Дальнейшая редукция со степенью 16 приведет к использованию одного устройства, выполняющего операции умножения, сложения и вычитания, что соответствует процедурной реализации на микропроцессоре.

Представленная иллюстрация принципов функционирования редукции производительности по вычислительным устройствам на примере операции быстрого преобразования Фурье показывает, что 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 – во сколько раз сокращается число функциональных устройств.

Другие виды редукции производительности также сокращают параметры реализации параллельной программы. Редукция производительности по каналам памяти представляет собой согласованное сокращение числа одновременно используемых каналов памяти для фрагмента информационного графа прикладной задачи.

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.

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. Согласование потоков данных для промежуточных вариантов реализации задачи, сочетающих схемотехническую реализацию фрагмента задачи на реконфигурируемом вычислительном узле ВСГТ и многопоточные вычисления в микропроцессорах общего назначения ВСГТ, осуществляется с помощью интерфейсов и элементов синхронизации, которые автоматически устанавливает программа-синтезатор из соответствующей библиотеки. 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.

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.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Ilya Levin, Alexey Dordopulo, Vasiliy Kovalenko, Viacheslav Gudkov, Andrey Gulenok*. Programming tools for reconfigurable computer systems based on Virtex-7 FPGAs with using soft-architectures // 13th International Conference on Parallel Computing Technologies (PaCT-2015), Petrozavodsk, Russia, August 31-September 4, 2015. – C. 349-362.
2. *Dong X, Chai J, Yang J, Wen M, Wu N, Cai X, Zhang C, Chen Z.* Utilizing multiple xeon Phi coprocessors 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. – 2014. – Vol. 8631 LNCS, Issue PART 2. – P. 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. – 1 January 2016. – Vol. 62. – P. 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. – May 2016. – Vol. 32, Issue 3. – P. 517-539.
5. *Евстигнеев Н.М., Рябков О.И.* Применение архитектуры multiGPU+CPU для задач прямого численного моделирования ламинарно-турбулентного перехода при рассмотрении задач в качестве нелинейных динамических систем // Параллельные вычислительные технологии (ПАВТ'2016). – Челябинск: Издательский центр ЮУрГУ, 2016. – С. 141-154.
6. *Dordopulo Aleksey, Levin Ilya, Kalyaev Igor, Gudkov Vyacheslav, Gulenok Andrey.* Programming of hybrid computer systems based on the performance reduction method // Parallel Computing Technologies (PCT 2016), Proceedings of the 10th Annual International Scientific Conference on Parallel Computing Technologies, Arkhangelsk, Russia, 2016. – P. 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. – P. 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. – P. 171-174.
9. *Городничев М.А., Дучков А.А., Сарычев В. Г.* Программная реализация метода когерентного суммирования на GPU с использованием программной модели NVIDIA CUDA // Параллельные вычислительные технологии (ПаВТ'2016). – URL: https://www.agora.guru.ru/pavt. – С. 118-130.
10. *Каляев И. А., Дордопуло А. И., Левин И. И., Гудков В. А., Гуленок А. А.* Технология программирования вычислительных систем гибридного типа // Вычислительные технологии. – 2016. – Т. 21, № 3. – С. 33-44. ISSN 1560-7534.
11. *Семерникова Е.Е., Левин И.И., Гудков В.А.* Организация битовой обработки данных для реконфигурируемых вычислительных систем на языке программирования высокого уровня // Вестник компьютерных и информационных технологий. – 2015. – № 5. – С. 3-9.
12. *Дордопуло А.И., Левин И.И., Каляев И.А., Гудков В.А., Гуленок А.А.* Параллельно-конвейерная форма программы как основа программирования вычислительных систем гибридного типа // Вестник УГАТУ. – 2016. – Т. 20 (73), № 3. – С. 122-128. ISBN 978-5-9275-1980-7.

13. *Danilov I.G., Dordopulo A.I., Kalyaev Z.V., Levin I.I., Gudkov V.A., Gulenok A.A. and Bovkun A.V.* Distributed Monitoring System For Reconfigurable Computer Systems // Procedia Computer Science. – 2016. – No. 101. – P. 341-350.
14. *Антонов А.С., Воеводин Вад В., Даугель-Дауге А.А., Жуматий С.А., Никитенко Д.А., Соболев С.И., Стефанов К.С., Швец П.А.* Обеспечение оперативного контроля и эффективной автономной работы Суперкомпьютерного комплекса МГУ // Вестник Южно-Уральского государственного университета. Серия "Вычислительная математика и информатика". – 2015. – Т. 4 (2). – С. 33-43.
15. *Мовчан А.В., Цымблер М.Л.* Параллельная реализация поиска самой похожей подпоследовательности временного ряда для систем с распределенной памятью // Параллельные вычислительные технологии (ПаВТ'2016). – Челябинск: Издательский центр ЮУрГУ, 2016. – С. 615-628.
16. *Баканов В.М.* Управление динамикой вычислений в процессорах потоковой архитектуры для различных типов алгоритмов // Программная инженерия. – 2015. – № 9. – С. 20-24.
17. *Konstantin Barkalov, Victor Gergel, Ilya Lebedev.* Use of Xeon Phi Coprocessor for Solving Global Optimization Problems // Parallel Computing Technologies. – 2015. – P. 307-318. DOI: 10.1007/978-3-319-21909-7_31.
18. *Konstantin Y. Besedin, Pavel S. Kostenetskiy, Stepan O. Prikazchikov.* Using Data Compression for Increasing Efficiency of Data Transfer Between Main Memory and Intel Xeon Phi Coprocessor or NVidia GPU in Parallel DBMS // Procedia Computer Science. – 2015. – Vol. 66. – P. 635-641.
19. *Bernard Goossens DALI, David Parello, Katarzyna Porada, Djallal Rahmoune.* Toward a Core Design to Distribute an Execution on a Manycore Processor // Proceedings of the 13th International Conference on Parallel Computing Technologies. – 2015. – Vol. 9251. – P. 390-404. ISBN: 978-3-319-21908-0. DOI: 10.1007/978-3-319-21909-7_38.
20. *Pavel Pavlukhin, Igor Menshov.* On Implementation High-Scalable CFD Solvers for Hybrid Clusters with Massively-Parallel Architectures // Lecture Notes in Computer Science. – 2015. – Vol. 9251. – P. 436-444.

REFERENCES

1. *Ilya Levin, Alexey Dordopulo, Vasiliy Kovalenko, Viacheslav Gudkov, Andrey Gulenok.* Programming tools for reconfigurable computer systems based on Virtex-7 FPGAs with using soft-architectures, *13th International Conference on Parallel Computing Technologies (PaCT-2015), Petrozavodsk, Russia, August 31-September 4, 2015,* pp. 349-362.
2. *Dong X, Chai J, Yang J, Wen M, Wu N, Cai X, Zhang C, Chen Z.* Utilizing multiple xeon Phi coprocessors 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,* 2014. – Vol. 8631 LNCS, Issue PART 2. – P. 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. 1 January 2016, Vol. 62, pp. 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,* May 2016, Vol. 32, Issue 3, pp. 517-539.
5. *Evstigneev N.M., Ryabkov O.I.* Primenenie arkhitektury multiGPU+CPU dlya zadach pryamogo chislennogo modelirovaniya laminarno-turbulentnogo perekhoda pri rassmotrenii zadach v kachestve nelineynykh dinamicheskikh sistem [Application architecture multiGPU + CPU tasks for direct numerical simulation of laminar-turbulent transition in considering problems as nonlinear dynamic systems], *Parallel'nye vychislitel'nye tekhnologii (PAVT'2016)* [Parallel computational technologies (PCT'2016)]. Chelyabinsk: Izdatel'skiy tsentr YuUrGU, 2016, pp. 141-154.
6. *Dordopulo Aleksey, Levin Ilya, Kalyaev Igor, Gudkov Vyacheslav, Gulenok Andrey.* Programming of hybrid computer systems based on the performance reduction method, *Parallel Computing Technologies (PCT 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*, pp. 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*, pp. 171-174.
9. *Gorodnichev M.A., Duchkov A.A., Sarychev V.G.* Programmnaya realizatsiya metoda kogerent-nogo summirovaniya na GPU s ispol'zovaniem programmnoy modeli NVIDIA CUDA [Software implementation of the method of coherent summation on the GPU with NVIDIA CUDA programming model], *Parallel'nye vychislitel'nye tekhnologii (PaVT'2016)* [Parallel computational technologies (PCT'2016)]. Available at: https://www.agora.guru.ru/pavt, pp. 118-130.
10. *Kalyaev I.A., Dordopulo A.I., Levin I.I., Gudkov V.A., Gulenok A.A.* Tekhnologiya programmirovaniya vychislitel'nykh sistem gibridnogo tipa [The technology of computer programming hybrid systems], *Vychislitel'nye tekhnologii* [Computational technologies], 2016, Vol. 21, No. 3, pp. 33-44. ISSN 1560-7534.
11. *Semernikova E.E., Levin I.I., Gudkov V.A.* Organizatsiya bitovoy obrabotki dannykh dlya rekonfiguriruemykh vychislitel'nykh sistem na yazyke programmirovaniya vysokogo urovnya [Organization of bit data processing for reconfigurable computer systems in the high-level programming language], *Vestnik komp'yuternykh i informatsionnykh tekhnologiy* [Herald of computer and information technologies], 2015, No. 5, pp. 3-9.
12. *Dordopulo A.I., Levin I.I., Kalyaev I.A., Gudkov V.A., Gulenok A.A.* Parallel'no-konveyernaya forma programmy kak osnova programmirovaniya vychislitel'nykh sistem gibridnogo tipa [In parallel-pipelined form of the program as a basis for programming hybrid computing systems], *Vestnik UGATU* [Bulletin of the South Ural State University], 2016, Vol. 20 (73), No. 3, pp. 122-128. ISBN 978-5-9275-1980-7.
13. *Danilov I.G., Dordopulo A.I., Kalyaev Z.V., Levin I.I., Gudkov V.A., Gulenok A.A. and Bovkun A.V.* Distributed Monitoring System For Reconfigurable Computer Systems, *Procedia Computer Science*, 2016, No. 101, pp. 341-350.
14. *Antonov A.S., Voevodin Vad V., Daugel'-Dauge A.A., Zhumatiy S.A., Nikitenko D.A., Sobolev S.I., Stefanov K.S., Shvets P.A.* Obespechenie operativnogo kontrolya i effektivnoy avtonomnoy raboty Superkomp'yuternogo kompleksa MGU [Providing run-time control and effective offline work of MSU Supercomputer complex], *Vestnik Yuzhno-Ural'skogo gosudarstvennogo universiteta. Seriya "Vychislitel'naya matematika i informatika"* [Bulletin of South Ural State University. Series: Computational mathemathics and informatics], 2015, Vol. 4 (2), pp. 33-43.
15. *Movchan A.V., Tsymbler M.L.* Parallel'naya realizatsiya poiska samoy pokhozhey podposledovatel'nosti vremennogo ryada dlya sistem s raspredelennoy pamyat'yu [Parallel implementation of the search itself like a subsequence of the time series for systems with distributed memory], *Parallel'nye vychislitel'nye tekhnologii (PaVT'2016)* [Parallel computational technologies (PCT'2016)]. Chelyabinsk: Izdatel'skiy tsentr YuUrGU, 2016, pp. 615-628.
16. *Bakanov V.M.* Upravlenie dinamikoy vychisleniy v protsessorakh potokovoy arkhitektury dlya razlichnykh tipov algoritmov [Management dynamic computing processors in a streaming architecture for various types of algorithms], *Programmnaya inzheneriya* [Software Engineering], 2015, No. 9, pp. 20-24.
17. *Konstantin Barkalov, Victor Gergel, Ilya Lebedev.* Use of Xeon Phi Coprocessor for Solving Global Optimization Problems, *Parallel Computing Technologies*, 2015, pp. 307-318. DOI: 10.1007/978-3-319-21909-7_31.
18. *Konstantin Y. Besedin, Pavel S. Kostenetskiy, Stepan O. Prikazchikov.* Using Data Compression for Increasing Efficiency of Data Transfer Between Main Memory and Intel Xeon Phi Coprocessor or NVidia GPU in Parallel DBMS, *Procedia Computer Science*, 2015. Vol. 66, pp. 635-641.
19. *Bernard Goossens DALI, David Parello, Katarzyna Porada, Djallal Rahmoune*. Toward a Core Design to Distribute an Execution on a Manycore Processor, *Proceedings of the 13th International Conference on Parallel Computing Technologies*, 2015, Vol. 9251, pp. 390-404. ISBN: 978-3-319-21908-0. DOI: 10.1007/978-3-319-21909-7_38.

20. *Pavel Pavlukhin, Igor Menshov.* On Implementation High-Scalable CFD Solvers for Hybrid Clusters with Massively-Parallel Architectures, *Lecture Notes in Computer Science*, 2015, Vol. 9251, pp. 436-444.

Статью рекомендовал к опубликованию д.т.н. А.С. Горобцов.

Дордопуло Алексей Игоревич – Южный федеральный университет; e-mail: scorpio@mvs.tsure.ru; 347928, г. Таганрог, 10-й пер. 114/1; тел. 88634361364; НИИ МВС ЮФУ; заведующий лабораторией; к.т.н.

Левин Илья Израилевич – e-mail: levin@superevm.ru; 347928, г. Таганрог, ул. Петровская, д. 15, кв. 143; тел. 88634612111; Научно-исследовательский центр супер-ЭВМ и нейрокомпьютеров; директор; д.т.н.; профессор.

Каляев Игорь Анатольевич – e-mail: kaliaev@mvs.sfedu.ru; 347928, г. Таганрог, ул. Р. Люксембург, 8а; тел. 88634360657; НИИ МВС ЮФУ; г.н.с.; чл.-корр. РАН; д.т.н.; профессор.

Гудков Вячеслав Александрович – e-mail: gudkov@mvs.tsure.ru; 347905, г. Таганрог, ул. Дзержинского, 110; тел. 88634612111; Научно-исследовательский центр супер-ЭВМ и нейрокомпьютеров; с.н.с.; к.т.н.

Гуленок Андрей Александрович – e-mail: andrei_gulenok@mail.ru; 347905, г. Таганрог, ул. Амвросиевская, 78; НИИ МВС ЮФУ; с.н.с.; к.т.н.

Dordopulo Alexey Igorevitch – Southern Federal University; e-mail: scorpio@mvs.tsure.ru; 114/1, 10th lane, Taganrog, 347928, Russia; phone +78634361364; SRI MCS SFU; head of laboratory; cand. of eng. sc.

Levin Ilya Izrailevitch – e-mail: levin@superevm.ru; 15, Petrovskaya street, ap. 143, Taganrog, 347928, Russia; phone +78634612111; Scientific Research Centre of Supercomputers and Neurocomputers; director; dr. of eng. sc.; professor.

Kalyaev Igor Anatolievitch – e-mail: kaliaev@mvs.sfedu.ru; 8a, Rosa Luxembourg street, Taganrog, 347928, Russia; phone +78634360657; SRI MCS SFU; chief research associate; correspondent member of the RAS; dr. of eng. sc.; professor.

Gudkov Vyacheslav Alexandrovitch – e-mail: gudkov@mvs.tsure.ru; 110, Dzerzhinskogo street, Taganrog, 347905, Russia; phone +78634612111; Scientific Research Centre of Supercomputers and Neurocomputers; senior staff scientist; cand. of eng. sc.

Gulenok Andrei Alexandrovitch – e-mail: andrei_gulenok@mail.ru; 78, Amvrosievskaya street, Taganrog, 347905, Russia; SRI MCS SFU; senior staff scientist; cand. of eng. sc.

УДК 004.272.26 DOI 10.18522/2311-3103-2016-11-5464

И.И. Кулагин, М.Г. Курносов

О СПЕКУЛЯТИВНОМ ВЫПОЛНЕНИИ КРИТИЧЕСКИХ СЕКЦИЙ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ НА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ С ОБЩЕЙ ПАМЯТЬЮ[*]

Целью работы является исследование метода выполнения критических секций при помощи программной транзакционной памяти, а также описание основных аспектов её реализации. Транзакционная память – это активно развивающийся примитив синхронизации, позволяющий создавать масштабируемые потокобезопасные параллельные программы для вычислительных систем с общей памятью. Для исследования была выбрана реализация транзакционной памяти в компиляторе GCC (библиотека libitm). Библиотека libitm

[*] Работа выполнена при финансовой поддержке РФФИ (гранты № 16-07-00712, 15-07-02693).