ПОВЫШЕНИЕ ЭФФЕКТИВНОСТИ ПРИМЕНЕНИЯ МЕТОДА СПЕКУЛЯТИВНОЙ МНОГОПОТОЧНОСТИ ДЛЯ ДИНАМИЧЕСКОГО РАСПАРАЛЛЕЛИВАНИЯ ПРОГРАММ

УДК 004.272.2

Марина Алексеевна Бакулева,
к.т.н., зав. каф. Информационных технологий Рязанского филиала Московского государственного университета экономики, статистики и информатики (МЭСИ)
Тел.: 8 (4912) 24-03-09
Эл. почта: marina.bakuleva@gmail.com

Александр Валериевич Бакулев,
к.т.н., доцент каф. Информационных технологий Рязанского филиала Московского государственного университета экономики, статистики и информатики (МЭСИ)
Тел.: 8 (4912) 24-03-09
Эл. почта: alex.bakulev@gmail.com

Рассмотрены возможности метода спекулятивной многопоточности для динамического распараллеливания последовательных программ. Осуществлена его формализации и определены характеристики метода, учёт которых позволит повысить эффективность его применения при автоматическом распараллеливании. Приведены результаты экспериментального исследования разработанного алгоритма динамического выделения спекулятивных регионов.

Ключевые слова: *многоядерные процессоры, параллельные алгоритмы, спекулятивная многопоточность, автоматическое распараллеливание.*

Marina A. Bakuleva,
PhD in technical, the head of the chair of Information Technology, Ryazan Department Moscow State University of Economics, Statistics and Informatics (MESI)
Tel.: 8 (4912) 24-03-09
E-mail: marina.bakuleva@gmail.com

Alexander V. Bakulev,
PhD in technical, Assistant Professor Department of Information Technology, Ryazan Department Moscow State University of Economics, Statistics and Informatics (MESI)
Tel.: 8 (4912) 24-03-09
E-mail: alex.bakulev@gmail.com

IMPROVING THE EFFICIENCY OF THE METHOD OF SPECULATIVE MULTI-THREADING FOR DYNAMIC PARALLELIZING

The possibilities of the method of speculative multithreading dynamic parallelization of sequential programs. It implemented its formalization and identified characteristics of the method, the account which will increase the efficiency of its use in automatic parallelizing. The results of experimental studies of the algorithm dynamically allocate speculative regions.

Keywords: *multi-core, parallel algorithms, speculative multithreading, automatic parallelization.*

1. Введение

Развитие технологий программирования для персональных компьютеров с точки зрения повышения производительности программ традиционно происходило в основном экстенсивными методами. По большей части разработчики программного обеспечения надеялись на постоянное совершенствование аппаратных ресурсов компьютера. Очередная замена процессора с увеличенной тактовой частотой позволяла получить ускорение работы программ без малейшего их изменения.

В настоящее время достигнут фактический предел роста тактовой частоты процессора, и дальнейшее увеличение его производительности становится возможным только за счет перехода к многоядерной архитектуре, по существу к параллельной архитектуре. На сегодняшний день уже широко распространены процессоры с четырьмя и более ядрами, а в перспективе количество ядер может измеряться сотнями, однако, чтобы обеспечить потенциал их производительности необходимо разрабатывать новые параллельные программы, обладающие необычными динамическими свойствами (масштабируемость на все доступные ресурсы, балансировка загрузки ядер, анализ зависимостей по данным и управлению и др.). Все это требует разработки новых подходов к параллельному программированию, которые направлены на упрощение и повышение эффективности работы программиста. Один из таких подходов предполагает добавление в арсенал современных компиляторов элементов *автоматического распараллеливания* существующих последовательных программ [1].

Цель работы – в работе представляются результаты исследования метода автоматического динамического распараллеливания последовательных программ – метода *спекулятивной многопоточности* [2]. Задачей такого исследования является необходимость строгой формализации данного метода, с целью определения основных параметров вычислительного процесса, расчёт характеристик которых позволит осуществлять автоматическое распараллеливание последовательных программ более эффективно, с меньшими накладными расходами.

Формализация метода. Суть метода спекулятивной многопоточности состоит в следующем. Среди множества всех регионов G последовательной программы выявляются регионы, имеющие зависимости по данным типа δ^{∞}_{out-in}, δ^{∞}_{in-out}, $\delta^{\infty}_{out-out}$, характер которых не может быть установлен на этапе трансляции программы из-за неоднозначности [3]. Статические методы при планировании параллельного исполнения подобных регионов, чтобы исключить нарушение информационных зависимостей в случае их возникновения, вынуждены использовать более сильные виды отношений, прибегая к позиции крайнего пессимизма [4]. Такой подход негативно отражается на результатах распараллеливания.

Метод спекулятивной многопоточности предписывает параллельное многопоточное выполнение подобных регионов, в расчёте на то, что информационные зависимости на стадии исполнения не проявятся, прибегая к позиции крайнего оптимизма (спекулируя на удаче). В случае если подобные надежды оправдают себя, будет получен выигрыш в производительности, в противном случае, результаты вычислений региона должны быть аннулированы, и он будет выполнен повторно, что приведет к накладным расходам.

Для того, чтобы организовать многопоточное выполнение спекулятивных регионов, формируется динамическая последовательность стадий их

исполнения, называемых *эпохами* $\forall e_i \in EP \mid i = \overline{1,N},\ N = |EP|$. Например, для циклического региона эпохами являются отдельные итерации цикла. Каждая эпоха снабжается локальным буфером памяти для сохранения критических к возможным нарушениям зависимости данных. Результаты выполнения эпохи приводятся в соответствие со следующими правилами.

1. После окончания спекулятивного исполнения каждой эпохи региона проверяется, не произошло ли при этом нарушения информационных зависимостей.

2. Если нарушения не было зафиксировано, результаты вычисления эпохи переносятся из защищённого буфера в общую память программы и эпоха освобождает свой процессорный элемент.

3. Если было зафиксировано нарушение информационной зависимости, результаты аннулируются и эпоха запускается на исполнение повторно.

4. Всем эпохам присваиваются уникальные номера. Нумерация эпох осуществляется строго в порядке их исполнения в последовательной программе. Таким образом, исполнение эпох в порядке следования их номеров гарантирует точное соответствие получаемых результатов результатам последовательного выполнения программы.

5. Проверка результатов выполнения эпохи на корректность осуществляется только после того, как эпоха с предыдущим номером успешно завершила своё выполнение.

6. Новые, ожидающие запуска на исполнение, эпохи распределяются по освободившимся от выполнения предыдущих эпох процессорным элементам строго в порядке следования их номеров.

Подобный подход позволяет фактически производить обработку отдельных итераций цикла в параллельном конвейерном режиме, заполняя ступени конвейера (процессорные элементы) новыми эпохами по мере их освобождения. Такой подход минимизирует возможные потери, даже в случае повторного выполнения неудачных эпох.

Учитывая правила 1–6, были сформулированы и доказаны [3] следующее утверждения.

Лемма 1. При одновременном параллельном спекулятивном исполнении p эпох $e_j, e_{j+1}, \ldots, e_{j+p-1}$, на вычислительной системе состоящей из p процессорных элементов, гарантируется корректность выполнения $\forall e_j$ и отсутствие её повторного перезапуска, где $j, j+1, \ldots, j+p-1$ – номера эпох, $e_j, e_{j+1}, \ldots, e_{j+p-1} \in EP \mid j = \overline{1,(N-p+1)}$.

Лемма 2. При одновременном параллельном спекулятивном исполнении p эпох $e_j, e_{j+1}, \ldots, e_{j+p-1} \in EP \mid j = \overline{1,(N-p+1)}$ на вычислительной системе состоящей из p процессорных элементов, корректность выполнения $\forall e_{j+k} \mid k = \overline{1,(p-1)}$ гарантируется, если подтверждена корректность $\forall e_{j+k-1}$ эпохи и проверки корректности результатов между каждой парой эпох $\forall (e_{j+k}, e_{j+m}) \mid m = \overline{0,(k-1)}$ дали положительный результат.

Лемма 3. Если при одновременном параллельном спекулятивном исполнении p эпох $e_j, e_{j+1}, \ldots, e_{j+p-1} \in EP \mid j = \overline{1,(N-p+1)}$ на вычислительной системе состоящей из p процессорных элементов $\exists e_{j+k} \mid k = \overline{1,(p-1)}$, для которой было зафиксировано нарушение корректности выполнения, то после перезапуска и повторного исполнения e_{j+k} её корректность будет гарантирована.

Сформулированные и доказанные леммы 1–3 позволили проанализировать работу метода спекулятивной многопоточности, ввести ряд параметров, выявить зависимости между ними, обосновать расчет важных характеристик, определяющих область его эффективной применимости.

2. Расчет характеристик метода

Поскольку наибольший объём вычислений в программах приходится на долю циклов, рассмотрим применение метода спекулятивной многопоточности для распараллеливания циклических регионов последовательной программы.

Пусть среди множества регионов программы выделено подмножество спекулятивных регионов $G^{SP} \subset G \mid G^{SP} = \{g_i\} \mid i = \overline{1,K}$, представляющих циклические управляющие операторы. Тогда для реализации их спекулятивного исполнения необходимо произвести развёртку итераций циклов в набор эпох, для чего зададим отношение $\rho_{EP} : G^{SP} \rightarrow EP$ между множеством спекулятивных регионов G^{SP} и множеством эпох EP. При этом требуется оценить перспективность такого разбиения, учитывая возможные издержки последующего спекулятивного исполнения регионов. Слишком короткие регионы (с малым числом операторов в теле цикла) необходимо разворачивать в набор последовательно выполняемых итераций, т.е. каждая эпоха такого региона будет состоять из нескольких смежных итераций цикла, что позволит увеличить время выполнения каждой эпохи, снизив долю постоянной составляющей накладных расходов.

Определим некоторые параметры спекулятивного региона $g_i \in G^{SP} \mid i = \overline{1,K}$ и выделенных на основе его циклических итераций эпох $\forall e_j \in EP_i \subseteq EP \mid j = \overline{1,N}, N = |EP_i|$.

Пусть n_δ есть число установленных зависимостей типа δ^{∞}_{out-in}, δ^{∞}_{in-out}, δ^{∞}_{out-ou}, между итерациями внутри региона g_i, $\Pr_{\delta-}$ – вероятность нарушения зависимости для одной итерации региона g_i; n_{It} – общее количество итераций тела циклического региона g_i; $n_{It/E}$ – число смежных итераций циклического региона g_i последовательно размещённых внутри каждой эпохи $\forall e_j \in EP_i$, является постоянным для каждого спекулятивного региона. Тогда общее количество эпох составит величину $\frac{n_{It}}{n_{It/E}}$, а общее число зависимостей между эпохами региона g_i будет составлять $n_{\delta/E} = n_\delta \cdot n_{It/E}$.

Пусть p – число процессорных элементов вычислительно системы, осуществляющих параллельное спекулятивное выполнение эпох региона g_i, а n_v – число операторов в теле цикла региона g_i.

Пусть $n_v(\varphi_{create})$ – количество операторов служебной подпрограммы, выполняющей функцию φ_{create} по созданию новых процессов на основе спекулятивных регионов. Пусть $n_v(\varphi_{writeback})$ – число операторов служебной подпрограммы, выпол-

няющей функцию $\varphi_{writeback}$ записи данных, защищенных от нарушения зависимостей δ^{∞}_{out-in}, δ^{∞}_{in-out}, δ^{∞}_{out-ou} с помощью локального буфера временной памяти эпохи, из этого буфера в общую память процессов (после того как исполнение данной эпохи будет признано корректным). Пусть $n_v(\varphi_{check})$ – число операторов служебной подпрограммы, выполняющей функцию φ_{check} проверки факта нарушения каждой из $n_{\delta/E}$ – зависимостей эпохи после её завершения.

Значения $n_v(\varphi_{create})$, $n_v(\varphi_{writeback})$ и $n_v(\varphi_{check})$ отражают системные издержки на реализацию механизма спекулятивного исполнения регионов.

Для того чтобы оценить возможные издержки на повторное исполнение неудачных эпох, определим среднее число нарушений зависимостей между эпохами на каждой фазе исполнения параллельного спекулятивного конвейера.

Рассмотрим процесс выполнения некоторой последовательности эпох e_j, e_{j+1},..., $e_{j+p-1} \in EP \mid j = \overline{1,(N-p+1)}$, $N = |EP|$ с момента запуска на исполнение до момента успешного завершения. В соответствии с леммой 2. число проверок корректности выполнения $\forall e_{j+k} \mid k = \overline{1,(p-1)}$ будет совпадать с $\forall (e_{j+k}, e_{j+m}) \mid m = \overline{0,(k-1)}$ количеством пар, т.е. составит величину k. Причем согласно лемме 3, если в ходе проверок будут обнаружены нарушения корректности e_{j+k} и эпоха будет перезапущена заново, проведения повторной проверки корректности не потребуется. Таким образом, $\forall(e_{j+k}, e_{j+m})$ проверку можно рассматривать как некоторое испытание со случайным исходом равным вероятности $\Pr_{\delta/e-}(e_{j+k}, e_{j+m})$ события «при выполнении эпохи e_{j+k} нарушается корректность информационных зависимостей относительно эпохи e_{j+m}». Причём все проверки являются независимыми друг от друга, и каждая способна завершится положительным или отрицательным результатом. Вероятность исхода такого события тогда можно выразить через вероятность $\Pr_{\delta-}$ (вероятность нарушения зависимости для одной итерации региона) и $n_{It/E}$ (число смежных итераций внутри эпохи) по теореме сложения вероятностей: $\Pr_{\delta/e-} = n_{It/E} \cdot \Pr_{\delta-}$. Так как для $\forall e_{j+k}$ потребуется совершить k таких проверок, каждая из которых является несовместным событием, а окончательный факт нарушения корректности исполнения эпохи e_{j+k} будет установлен, если хотя бы одно событие состоится, то вероятность события «корректность выполнения эпохи e_{j+k} нарушается» можно определить по теореме сложения вероятностей: $\Pr_{\delta/E-}(e_{j+k}) = k \cdot \Pr_{\delta/e-} = k \cdot n_{It/E} \cdot \Pr_{\delta-}$. Эту вероятность можно также трактовать, как математическое ожидание числа появлений события «нарушение корректности выполнения эпохи e_{j+k}» для единственного испытания, проводимого после окончания выполнения $\forall e_{j+k}$ эпохи. Вероятностная оценка общего числа нарушений корректности, которые могут возникнуть за время исполнения p эпох e_j, e_{j+1}, …, e_{j+p-1} можно определить тогда как сумму математических ожиданий по каждой из эпох $\forall e_{j+k} \mid k = \overline{1,(p-1)}$:

$$n_s = \sum_{k=1}^{p-1} \Pr_{\delta/E-}(e_{j+k}) =$$

$$= \sum_{k=1}^{p-1} k \cdot n_{It/E} \cdot \Pr_{\delta-} = n_{It/E} \cdot \Pr_{\delta-} \cdot \sum_{k=1}^{p-1} k =$$

$$= n_{It/E} \cdot \Pr_{\delta-} \cdot \frac{1+p-1}{2} \cdot$$

$$\cdot (p-1) = n_{It/E} \cdot \Pr_{\delta-} \cdot \frac{p \cdot (p-1)}{2}.$$

Оценим общий объём вычислений и накладных издержек на организацию спекулятивного выполнения итераций циклического региона g_i, представленных в виде параллельных эпох. Будем измерять величину объёма вычислений в количестве операций, в расчете на одну эпоху. В оценку войдут следующие составляющие.

1. Создание и организация спекулятивного исполнения каждой эпохи региона g_i потребует выполнения дополнительных операций независимо от успешного или неудачного завершения спекуляции. Таким образом, объём постоянных издержек составит величину:

$$V_{const} = n_v(\varphi_{create}) + n_v(\varphi_{writeback}) + n_\delta \cdot n_{It/E} \cdot n_v(\varphi_{check}).$$

2. Непосредственное исполнение каждой эпохи потребует проведения некоторого объёма полезных вычислений (собственно ради чего и планируются параллельное выполнение):

$$V_{const} = n_{It/E} \cdot n_v.$$

3. Повторный перезапуск и выполнение эпохи, с учётом вероятности неудачного завершения спекуляции, составит следующий объём переменных издержек:

$$V_{\text{var}} = V_{comp} \cdot \frac{n_s}{p} = n_{It/E} \cdot n_v \cdot$$

$$\cdot \frac{n_{It/E} \cdot \Pr_{\delta-} \cdot p \cdot (p-1)}{2p} =$$

$$= 1/2 \cdot n^2_{It/E} \cdot n_v \cdot \Pr_{\delta-} \cdot (p-1).$$

Таким образом, общий объём вычислений, производимый при спекулятивном выполнении всех эпох региона g_i, измеренный в количестве операторов составит:

$$V_{all}(g_i) = \frac{n_{It}}{n_{It/E}} \cdot \left(V_{comp} + V_{const} + V_{\text{var}}\right) =$$

$$= \frac{n_{It}}{n_{It/E}} \cdot \begin{pmatrix} n_{It/E} \cdot n_v + n_v(\varphi_{create}) + \\ + n_v(\varphi_{writeback}) + \\ + n_\delta \cdot n_{It/E} \cdot n_v(\varphi_{check}) + \\ + 1/2 \cdot n^2_{It/E} \cdot n_v \cdot \Pr_{\delta-} \cdot (p-1) \end{pmatrix}.$$

Учтём параллельный характер вычислений, определив удельный объём вычислений, выполняемый одним процессором при спекулятивном выполнении региона g_i на вычислительной системе с p процессорными элементами:

$$V_{par}(g_i) = \frac{V_{all}(g_i)}{p}.$$

Удельный объём будет характеризовать ускорение, получаемое за счёт параллельной спекуляции. Так, при последовательном выполнении региона g_i потребовалось бы произвести следующий объём вычислений:

$$V_{seq}(g_i) = n_{It} \cdot n_v.$$

Тогда выигрыш от использования параллельной спекуляции можно измерить, соотнеся $V_{par}(g_i)$ и $V_{seq}(g_i)$:

$$n_{spd}(g_i) = \frac{V_{seq}(g_i)}{V_{par}(g_i)} = \\ = p \Big/ \left(\begin{array}{c} 1 + 1/2 \cdot n_{It/E} \cdot \Pr_{\delta-} \cdot (p-1) + \\ \left(\begin{array}{c} n_v(\varphi_{create}) + \\ + n_v(\varphi_{writeback}) + \\ + n_\delta \cdot n_{It/E} \cdot n_v(\varphi_{check}) \end{array} \right) \Big/ \\ / n_v \cdot n_{It/E} \end{array} \right).$$

Выведенная зависимость позволяет учитывать как особенности распараллеливаемой программы, так и возможности целевой вычислительной системы на которой планируется динамическое спекулятивное параллельное исполнение циклов программы. Важно отметить то, что $n_{spd}(g_i)$ не зависит от числа итераций цикла, что позволяет производить оценку перспективности распараллеливания циклических участков с заранее неизвестным числом повторений.

Заданную зависимость можно использовать в качестве основного критерия оптимальности при формировании спекулятивных регионов и эпох из последовательной программы в процессе автоматического динамического спекулятивного многопоточного распараллеливания.

3. Алгоритм выделения спекулятивных регионов и эпох

Для использования метода спекулятивной многопоточности при организации динамического исполнения параллельного промежуточного кода на вычислительной системе с однородными параллельными процессорными элементами и общей памятью, необходимо разработать способ определения спекулятивных регионов, выделения на их основе эпох и оценки выигрыша от последующей спекуляции.

В качестве основного критерия при выборе спекулятивных регионов и эпох будем использовать зависимость $n_{spd}(g_i)$. Как следует из анализа выражения $n_{spd}(g_i)$, данные действия можно осуществлять только на стадии исполнения программы, когда известны параметры целевой вычислительной системы: количество доступных процессорных элементов p и оценки трудоёмкости выполнения служебных процедур $n_v(\varphi_{create})$, $n_v(\varphi_{writeback})$, $n_v(\varphi_{check})$.

Вместе с тем, параметры циклических регионов могут быть рассчитаны ещё на стадии трансляции последовательной программы в промежуточный код: количество зависимостей типа δ^{∞} между операторами внутри региона n_δ, число операторов в теле цикла региона n_v, вероятность нарушения хотя бы одной из n_δ-зависимостей для всего региона $\Pr_{\delta-}$. Это позволит сократить накладные расходы при динамическом планировании параллельных регионов во время исполнения программы.

Таким образом, оказываются известны все параметры выражения $n_{spd}(g_i)$, кроме числа смежных итераций цикла приходящихся на каждую спекулятивную эпоху $n_{It/E}$. Именно этот параметр необходимо варьировать с целью максимизации значения выражения $n_{spd}(g_i)$. При этом общим ограничением является условие целесообразности организации спекулятивного распараллеливания: $n_{spd}(g_i) > 1$.

С учётом данных положений предлагается следующий алгоритм выделения спекулятивных регионов и эпох на основе промежуточного кода.

Вход. Множество регионов $G = \{g_i\} \mid i = \overline{1,N}$ промежуточного кода программы. Параметры целевой вычислительной системы: p, $n_v(\varphi_{create})$, $n_v(\varphi_{writeback})$, $n_v(\varphi_{check})$. Параметры $\forall g_i$: $n_v(g_i)$, $n_\delta(g_i)$, $\Pr_{\delta-}(g_i)$, $i = 0$.

Шаг 1. $i = i + 1$.

Если $i > N$,

то переход на *шаг 8*.

Шаг 2. Определяем тип стража c_j управляющего оператора, с которым связан регион g_i.

Шаг 3. Если $c_j \neq c^{mc}$,

то переход на *шаг 1*.

Шаг 4. Определяем значение $n^*_{It/E}(g_i)$, для которого показатель:

$$n^*_{spd}(g_i) \to \max_{n_{It/E}(g_i)} \left\{ n_{spd}(g_i) \right\}.$$

Шаг 5. Если $n^*_{spd}(g_i) \leq 1$,

то переход на *шаг 1*.

Шаг 6. Добавляем новый регион g_i к множеству спекулятивных регионов G^{SP}, так что теперь $g_i \in G^{SP}$. Задаём для g_i значение показателя $n_{It/E}(g_i) = n_{It/E}{}^*(g_i)$.

Шаг 7. Переходим на *шаг 1*.

Шаг 8. Конец алгоритма.

Множество спекулятивных регионов G^{SP} сформировано.

Предложенный алгоритм позволяет сформировать множество спекулятивных регионов G^{SP} за один просмотр списка регионов параллельного промежуточного кода и имеет трудоёмкость порядка $O(N)$, где $N = |G|$.

На основе множества регионов G^{SP} можно обеспечить динамическое планирование выполнения последовательной программы по методу спекулятивной многопоточности.

4. Экспериментальные исследования

Для оценки эффективности работы метода спекулятивной многопоточности были проведены эксперименты. В рамках экспериментов было организовано многопоточное исполнение кода двух программ: архивации файлов по методу LZW (LZW) и реализации алгоритмов оперативного анализа, основанных на кратномасштабном представлении данных (OLAP). Все программы

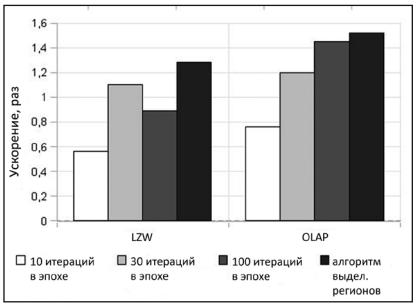

Рис. 1. Результаты ускорения многопоточного выполнения программ (Win)

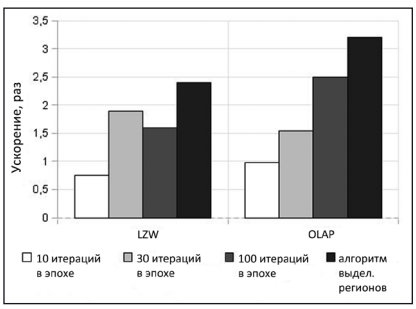

Рис. 2. Результаты ускорения многопоточного выполнения программ (Linux)

неоднократно прогонялись на различных наборах данных.

Для того, чтобы оценить влияние предложенного алгоритма выделения спекулятивных регионов и эпох на эффективность использования метода спекулятивной многопоточности, было выполнено сравнение результатов спекулятивного многопоточного выполнения разных версий кода программ LZW и OLAP с фиксированным количеством смежных итераций в каждой спекулятивной эпохе (10, 30 и 100) и выделенным динамически с помощью алгоритма. Результаты ускорения от использования метода спекулятивной многопоточности для двухъядерной системы на базе Intel/Windows представлены на рисунке 1, для 4-х ядерной системы на базе AMD/Linux представлены на рисунке 2.

5. Заключение

В статье рассмотрены возможности использования метода спекулятивной многопоточности для автоматического распараллеливания последовательных программ. Данный метод динамического распараллеливания позволяет существенно упростить анализ зависимостей по данным и управлению между последовательными регионами таких программ, повысить за счёт этого возможности их параллельного исполнения.

Использование предложенного алгоритма позволило повысить производительность параллельной обработки метода спекулятивной многопоточности от 5% до 65% за счет рационального выделения спекулятивных регионов и определения оптимального количества смежных итераций, выполняемых внутри каждой спекулятивной эпохи.

Литература

1. Корячко В.П., Скворцов С.В., Телков И.А. Архитектуры многопроцессорных систем и параллельные вычисления. – М.: Высшая школа, 1999. 235 с.

2. Steffan J.G., Mowry T.C. The Potential for Using Thread-Level Data Speculation to Facilitate Automatic Parallelization. Carnegie Mellon University, HPCA-4, February 1–4, 1998.

3. Bakulev A.V., Bakuleva M.A., Avilkina S.B. Mathematical methods and algorithms of mobile parallel computing on the base of multi-core processors // European researcher. 2012. V. 33. № 11-1. P. 1826–1834.

4. Бакулев А.В., Бакулева М.А., Козлов М.А., Скворцов С.В. Технологии разработки параллельных программ для современных многоядерных процессоров. //Экономика, статистика и информатика. Вестник УМО. 2014. № 6. С. 211–215.

References

1.Koryachko V.P., Skvortsov S.V., Telkoff I.A. Architecture multiprocessor systems and parallel-calculations. – M .: Higher School, 1999. 235p.

2. Steffan J.G., Mowry T.C. The Potential for Using Thread-Level Data Speculation to Facilitate Automatic Parallelization. Carnegie Mellon University, HPCA-4, February 1–4, 1998.

3. Bakulev A.V., Bakuleva M.A., Avilkina S.B. Mathematical methods and algorithms of mobile parallel computing on the base of multi-core processors // European researcher. 2012. V. 33. № 11-1. P. 1826–1834.

4. Bakulev A.V., Bakuleva M.A., Skvortsov S.V., Kozlov M.A. Parallel Programming for modern multi-core processors// Economics, Statistics and Informatics. Bulletin of UMO. 2014. № 6. P. 211–215.