УДК 004.27 DOI 10.31114/2078-7707-2018-3-10-15

Методы планирования вычислений в параллельной потоковой вычислительной системе «Буран»

Д.Н. Змеев, А.В. Климов, Н.Н. Левченко, А.С. Окунев

Институт проблем проектирования в микроэлектронике РАН, г. Москва, zmejevdn@ippm.ru, klimov@ippm.ru, nick@ippm.ru, oku@ippm.ru

***Аннотация* — Параллельная потоковая вычислительная система «Буран» реализует потоковую модель вычислений с динамически формируемым контекстом. Для обеспечения эффективного распараллеливания задач на большое число ядер системы необходимо управлять вычислительным процессом в автоматическом режиме и решать задачи с большим объемом данных, имея ограниченный объем локальной памяти вычислительного ядра. В статье описываются методы планирования вычислений, которые позволяют преодолеть, а в ряде случаев и исключить саму возможность переполнения ассоциативной памяти ключей вычислительного ядра системы. Такими методами являются: метод с использованием токенов-заглушек; метод, использующий разбиение токенов на этапы (активные и пассивные) и откачку пассивных этапов в соответствующую память; метод с использованием только отложенных этапов (без откачки). Эти методы повышают эффективность решения задач и увеличивают надежность работы вычислительных средств.**

***Ключевые слова* — планирование вычислений, параллельная потоковая вычислительная система, разбиение на этапы, потоковая модель вычислений.**

I. Введение

Высокопроизводительные вычислительные системы кластерного типа, демонстрирующие низкую реальную производительность на большом круге актуальных задач, по словам специалистов [1] нуждаются в изменении модели вычислений. Задача изменения модели вычислений актуальна также для разрабатываемых суперкомпьютерных систем, которые всегда были и остаются передовым фронтом развития вычислительной техники.

Примером такой модели вычислений, применение которой может быть актуальным при разработке суперкомпьютеров [2], является оригинальная модель вычислений с динамически формируемым контекстом. Работа по развитию этой модели вычислений и созданию архитектуры, ее реализующей (ППВС «Буран»), ведется в ИППМ РАН. Данная модель вычислений обладает рядом преимуществ по сравнению с применяемыми в традиционных вычислительных системах [3]-[4].

ППВС (параллельная поток вычислительная система) «Буран» [5]-[7] представляет собой многоядерную масштабируемую вычислительную систему. В состав вычислительного ядра этой системы входят исполнительное устройство, процессор сопоставления, коммутатор токенов, блок хэширования. Между ядрами в системе передаются единицы информации (в виде токенов), представляющие собой сообщения, содержащие помимо операнда набор служебных полей и тег – ключ. Коммутация между ядрами осуществляется по номеру вычислительного ядра, который вырабатывается в блоке хэширования настраиваемой программистом функцией распределения на основе полей токена.

Процессор сопоставления (ПС) [8] в ППВС обеспечивает взаимодействие токенов различных типов в зависимости от содержимого поля токена «код операции», сравнивая поля ключей токенов с учетом кратности и оценивая состояние других полей токенов. Ключи токенов сравниваются в ассоциативной памяти ключей (АПК), а сами токены хранятся в памяти токенов (ПТ). Память ключей и память токенов не могут быть бесконечного размера.

Для того, чтобы обеспечить работу программ в ППВС, необходимо эффективно управлять заполнением локальной памяти процессора сопоставления вычислительного ядра. Для этого были разработаны методы планирования вычислений, которые позволяют снижать активность вычислительных процессов, освобождая память от некоторых групп токенов, либо активизировать работу, возвращая токены в память сопоставлений.

В 80-90-х годах XX века велись работы над классическими потоковыми вычислительными системами в США, Великобритании и Японии [9]-[10], которые представляли собой по большей части вычислительные системы для выполнения традиционных (обычных) программ, но в новой модели вычислений.

Однако, в силу того, что не был решен целый ряд существенных вопросов – таких как регулирование параллелизма, распределение вычислений во времени и по пространству, построение иерархии ассоциативной памяти, а также из-за ограничений, связанных с уровнем развития существовавшей на тот момент элементной базы, данное направление было свернуто.

В последнее время исследователи все чаще начинают проявлять интерес к вычислениям с управлением потоком данных. Современными проектами, в модели вычислений которых используется принцип потока

данных (в основном «static dataflow architectures»), являются WaveScalar, TRIPS, Merrimac и ряд других. За рубежом, например, ведется разработка вычислительной системы «Maxeler», в которой ускоритель построен на принципах статического dataflow [11].

II. Средства планирования вычислений

Проблемы, которые не были решены ранее при проектировании систем, реализующих модель вычислений с управлением потоком данных, исследовались и решались при создании ППВС «Буран».

В частности, такими проблемами являлись:

- необходимость балансировки загрузки вычислительных ядер системы;
- снижение нагрузки на коммуникационную сеть, обмен в которой осуществляется «короткими» сообщениями;
- планирование вычислений.

Средства планирования вычислений в ППВС в общем случае можно разделить на:

- распределение вычислений по пространству (локализация по вычислительным ядрам);
- распределение вычислений во времени;
- управление с помощью различных методов ввода данных в систему.

Под термином «планирование вычислений» мы в первую очередь понимаем управление вычислительным процессом в «автоматическом» режиме, то есть возможность прохождения больших задач в условиях ограниченного объёма ассоциативной памяти ключей.

Возникающее в процессе вычислений переполнение АПК может привести к блокировке вычислительного процесса, что и определяет актуальность разрабатываемых аппаратно-программных методов планирования вычислений. В данной статье описываются методы распределения вычислений во времени: с использованием откачки и подкачки токенов, с использованием откачки и подкачки «активных» и «пассивных» этапов и с использованием только «отложенных» этапов.

III. Метод планирования вычислений с использованием откачки и подкачки токенов в динамическом режиме

Работа метода откачки/подкачки токенов основывается на специальном токене-заглушке, который функционирует следующим образом. Поступая в АПК с бесконечной кратностью, токен-заглушка все сопоставившиеся с ним токены направляет на откачку, а сам остается в АПК дожидаться прихода остальных токенов диапазона. Вновь пришедшие токены откачиваемого диапазона в АПК будут сопоставляться с токеном-заглушкой и откачиваться. Останавливать вычислительный процесс нет необходимости, так как работа идет в стандартном режиме. Программист определяет диапазоны откачиваемых токенов путем выбора соответствующего ключа и маски токена-заглушки.

В аппаратуру ПС вводится отдельный блок сбора статистики, который собирает информацию о числе токенов по каждому из диапазонов, а также по загрузке АПК. Также программист определяет пороги срабатывания откачки и подкачки: верхний порог – по достижении этого числа токенов в АПК начнется откачка одного из диапазонов токенов; и нижний порог – по достижении этого значения начнется подкачка одного из ранее откачанных диапазонов токенов (рис. 1). По умолчанию в качестве значения порогов устанавливается некоторая усредненная величина, определяемая по результатам прохождения задач.

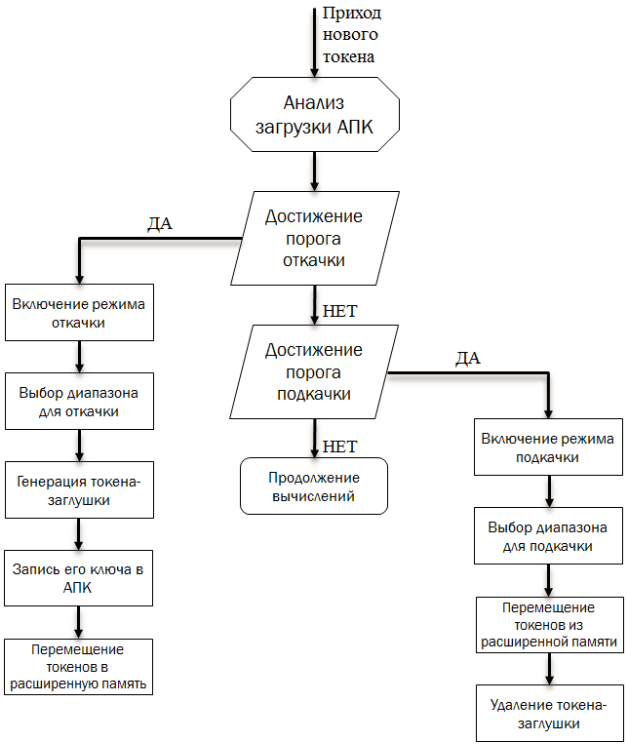

Рис. 1. Алгоритм функционирования метода откачки и подкачки токенов

Таким образом, на основе информации о загрузке АПК принимается решение об откачке или подкачке диапазона токенов. Выбор откачиваемого и подкачиваемого диапазона осуществляется либо по приоритету, либо по размеру – числу токенов в диапазоне. В случае если откачки одного диапазона недостаточно для продолжения вычислений, то принимается решение об откачке следующего диапазона.

При достижении АПК нижнего порога начинает работать механизм подкачки диапазона токенов.

Преимущества данного метода заключаются в следующем: простая аппаратная поддержка данного метода (введение токена-заглушки вполне достаточно для обеспечения механизма функционирования данного метода); отсутствие приостановки вычислительного процесса во время откачки и подкачки; уменьшение требуемого размера АПК для прохождения задач. Не-

достатками метода являются: сложность определения диапазона токенов не кратного числу 2, поскольку он определяется побитовым ключом с маской; сложность в назначении пороговых величин в АПК для откачки и подкачки токенов, так как в некоторых ситуациях неверный выбор порогов может привести к резкому возрастанию числа обменов с памятью («трежинг»).

Данный метод не способен полностью решить проблему переполнения АПК. Кроме того, при создании АПК большого объема при переходе на следующий уровень иерархии теряется возможность ассоциативного поиска на всех последующих уровнях, то есть токены откачиваются как бы в «пассивную» память по командам из устройства управления, откуда их обязательно нужно в какой-то момент вернуть в работу для последующего сопоставления. Кроме того, требуется дополнительная информация от программиста (компилятора) для порождения «заглушек».

Еще одна проблема данного метода заключается в том, что при возврате токена в ПС необходимо осуществить взаимодействие его со всеми или с частью находящихся там токенов, причем уже неизвестно, с какими надо взаимодействовать, а с какими нет. Поэтому для реализации этого механизма требуется, чтобы любой токен, лежащий в ПС, можно в любой момент изъять и заново направить на вход ПС. Это требование соблюдается, если при любом взаимодействии хотя бы один из участвующих во взаимодействии токенов удаляется из ПС.

IV. Метод планирования вычислений с использованием откачки и подкачки активных и пассивных этапов

Другим методом планирования вычислений является распределение во времени с использованием временных этапов. Временным этапом называется группа токенов, которая используется при выполнении определенного интервала вычислений – этапа. Обычно такая группа токенов связана с довольно обособленной частью программы (например, итерации в программе могут быть оформлены как временные этапы).

На этапы задача разбивается программистом путем применения хэш-функций, которые можно настроить непосредственно под выполняемую задачу при помощи различных параметров. Хэш-функции в процессе выполнения задачи присваивают этапам определенные номера, с которыми в дальнейшем и работает соответствующая аппаратура, «размещая» эти этапы во времени прохождения задачи. Все этапы делятся на «активные», токены которых обрабатываются в АПК, и «пассивные», токены которых приостанавливаются (задерживается их обработка) в ПС. Группа «пассивных» этапов состоит из «отложенных» и «откачанных». Ключи токенов «отложенного» этапа удаляются из АПК, но сами токены остаются в памяти токенов ПС, также «отложенные» этапы имеют приоритет при подкачке над «откачанными». Токены «откачанного» этапа перемещаются из памяти токенов (ПТ) в расширенную память.

К процедуре разбиения на этапы и к размеру этапа имеется ряд требований:

- этапы с большими номерами не должны генерировать токены с меньшими номерами этапов;
- размер этапа не должен превышать 10% от объема АПК ПС;
- одновременно «активными» могут быть 6-7 этапов, после завершения обработки одного из них происходит активация следующего и т.д.

Каждый токен, поступая на вход ПС, проходит процедуру хэширования, после которой ему присваивается номер этапа, на котором он будет востребован. После этого происходит проверка, к какому этапу относится токен. Если к «активному» – следует стандартная работа, токен подается в АПК на обработку, если к «пассивному» - токен откладывается. Если уровень заполненности АПК превосходит порог откачки, то происходит деактивация одного из активных этапов этапа (с наибольшим номером) и его токены откачиваются. При освобождении места в АПК для выполнения «отложенного» этапа происходит его активация.

Метод может функционировать и без задания порогов откачки. В этом случае откачка этапов происходит в автоматическом режиме при заполнении АПК до уровня, например, 85-90% от всего объема АПК в соответствии с имеющимися приоритетами задач или номерами этапов.

На уровне аппаратуры для функционирования данного метода добавляется блок разбиения на этапы – блок хэширования, таблица этапов (ТЭ), память отложенных токенов и память откачанных токенов.

Таблица этапов предназначена для хранения состояния этапов. Каждая строка ТЭ состоит из трех полей: поля «№ этапа», поля «Статус этапа» и поля «Число токенов этапа». Поле «Число токенов этапа» требуется для сбора статистики. Если токен принадлежит к «отложенному» или «откачанному» этапу, то сразу же происходит изменение поля «Число токенов этапа». В противном случае, когда токен принадлежит к «активному» этапу, изменение данного поля происходит только по результатам обработки ключа токена в АПК, поскольку в случае взаимодействия пришедшего токена («верхнего») с уже имеющимися в АПК («нижними») может произойти уменьшение количества токенов этапа. Этап считается пустым, при счетчике токенов данного этапа равном нулю.

Основные критерии выбора этапа, имеющего приоритет перед другими этапами для его активации, следующие:

- номер наименьший из всех номеров этапов;
- у этого этапа – более высокая активность по токенам и/или пакетам;
- у этого этапа большее количество отложенных токенов.

Основные критерии выбора этапа для его деактивации:

- номер этапа наибольший;
- низкая активность по токенам и/или пакетам.

На первой стадии этап переводится в статус «отложенный». Из АПК удаляются все ключи выбранного этапа, токены же этого этапа остаются в ПТ. Если кроме АПК переполняется и ПТ, то этап переходит в статус «откачанный», и токены из ПТ откачиваются во внешнюю память. Операция откачки этапа завершается, когда токены откачиваемого этапа в ассоциативной памяти ключей отсутствуют. При освобождении памяти токенов этап переходит в статус «отложенный» и его токены подкачиваются в ПТ. В дальнейшем этап активируется.

Перевод этапов из активного состояния в пассивное и наоборот осуществляет аппаратура ПС. Этот метод более универсален, чем предыдущий, но требует увеличение количества оборудования и дополнительных усилий от программиста при разбиении задачи на этапы.

V. Метод планирования вычислений с использованием только отложенных этапов

Третий метод планирования вычислений является в некотором роде развитием предыдущего, в котором хоть и задавался требуемый размер одного этапа, но аппаратура могла нивелировать некоторую погрешность путем откачки «лишнего» этапа. В методе с использованием только «отложенных» этапов было решено отказаться от «откачанных» этапов и вообще – от самой процедуры откачки. При использовании этого метода откачиваться этапы при угрозе переполнения АПК уже не будут, поскольку угроза переполнения АПК сведена к минимуму. Таким образом в вычислениях будут участвовать только токены «активных» этапов и токены «отложенных» этапов.

Таблица этапов заменяется на таблицу активных этапов (ТАЭ). Таблица активных этапов работает по ассоциативному принципу. В ней содержатся номера только «активных» этапов.

Пришедший на вход ПС токен проходит процедуру получения в блоке хэширования номера этапа, к которому он относится. После этого токены этапов, которые не являются активными, то есть их номера этапов отсутствуют в ТАЭ, а ключи – в АПК, остаются на входе в АПК – в памяти отложенных этапов (ПОЭ). Память отложенных этапов имеет иерархическое построение. В эту память перемещаются токены этапов, которые не попали в ТАЭ. В ПОЭ токены попадают в любом порядке, но внутри памяти структурируются по номерам этапов. Если же пришедший на вход токен относится к «активному» этапу, то его ключ сразу направляется на поиск в АПК и далее следует стандартная работа системы. При переводе «отложенных» этапов в статус «активных» сначала выбираются токены этапов с меньшими номерами, а затем – с большими.

Требования к размеру этапа остаются такими же – порядка 10% от размера АПК. Пользователь может сам ограничить число одновременно работающих «активных» этапов, либо предоставить этот выбор аппаратуре. Как только один из этапов завершает работу, то следующий по номеру «отложенный» этап переводится в статус «активных» и его токены поступают на вход АПК.

В данном методе устанавливается только порог подкачки или порог активации. На практике порог активации может представлять собой число одновременно выполняющихся «активных» этапов. Порог же подкачки может использоваться и как дополнительный механизм, который анализируя загрузку АПК может принять решение о повышении числа «активных» этапов. «Отложенные» этапы могут переходить в разряд активных по сигналам управления из блока управления процессора сопоставления.

VI. Заключение

В статье представлены основные методы планирования вычислений, разработанные для ППВС «Буран». Реализация этих методов в составе программно-аппаратных средств ППВС позволяет решить проблему переполнения ассоциативной памяти ключей процессора сопоставления, что исключает блокировку вычислительного процесса при прохождении задач.

Основные алгоритмы функционирования аппаратных решений были проверены на программной модели ППВС. Можно сделать вывод о том, что основным прикладным методом планирования вычислений в ППВС является их локализация во времени с помощью задаваемых пользователем хеш-функций. Механизм ограничения числа одновременно присутствующих в АПК токенов какой-либо программы в системе может быть реализован разными методами.

Первый метод планирования вычислений ограничен из-за использования диапазона откачиваемых токенов (определяемый ключом с маской), который будет всегда кратен двум. Однако при этом первый метод не требует вычисления хэш-функции.

Второй метод можно считать наиболее универсальным методом планирования вычислений во времени, так как он объединяет работу с «отложенными» и с «откачанными» этапами, однако он требует усложнения аппаратуры, в частности, введения блока хэширования ключей токенов.

Третий метод имеет более простой алгоритм работы, и при этом будет обеспечиваться непрерывность вычислительного процесса с минимальными потерями в производительности, хотя хэширование ключей токенов также потребуется.

Применение описанных в статье методов позволяет системе управлять вычислительным процессом в автоматическом режиме и решать задачи с большим объемом данных, имея ограниченный объем локальной памяти вычислительного ядра.

Данная работа выполнена при частичной финансовой поддержке грантов РФФИ 17-07-00478 и 17-07-00324.

ЛИТЕРАТУРА

[1] Донгарра Дж. Экзафлопсное будущее суперкомпьютеров // Суперкомпьютеры. №1(1). 2010. С. 21–23.

[2] Стерлинг Т. Многоточие Стерлинга // Суперкомпьютер. №3. 2010. С. 17-20.

[3] Климов А.В., Левченко Н.Н., Окунев А.С. Преимущества потоковой модели вычислений в условиях неоднородных сетей // Журнал «Информационные технологии и вычислительные системы». 2012. № 2. С. 36-45.

[4] Левченко Н.Н., Окунев А.С., Стемпковский А.Л. Использование модели вычислений с управлением потоком данных и реализующей ее архитектуры для систем эксафлопсного уровня производительности // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). М.: ИППМ РАН, 2012. № 1. С. 459-462.

[5] Стемпковский А.Л., Левченко Н.Н., Окунев А.С., Цветков В.В. Параллельная потоковая вычислительная система – дальнейшее развитие архитектуры и структурной организации вычислительной системы с автоматическим распределением ресурсов // Журнал «ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ». 2008. №10. С. 2-7.

[6] Климов А.В., Левченко Н.Н., Окунев А.С., Стемпковский А.Л. Вопросы применения и реализации потоковой модели вычислений // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). М.: ИППМ РАН, 2016. № 2. С. 100-106.

[7] Змеев Д.Н., Левченко Н.Н., Окунев А.С., Стемпковский А.Л. Принципы организации системы ввода/вывода параллельной потоковой вычислительной системы // Программные системы: теория и приложения. 2015. 6:4(27). С. 3–28. URL: http://psta.psiras.ru/read/psta2015_4_3-28.pdf (дата обращения: 20.04.2018)

[8] Левченко Н.Н., Окунев А.С., Яхонтов Д.Е. Исследование работы процессора сопоставления параллельной потоковой вычислительной системы «Буран» // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). М.: ИППМ РАН, 2012. № 1. С. 467-470.

[9] Lee B., Hurson A.R. Dataflow Architectures and Multithreading // Computer. 1994. Aug. V. 27. № 8. P. 27-39.

[10] Silc J., Robic B., Ungerer T. Asynchrony in parallel computing: From dataflow to multithreading // Parallel and Distributed Computing Practices. 1998. Vol. 1. № 1. P. 3-30.

[11] Soukaina N.Hmid, Jose G.F. Coutinho, and Wayne Luk. A Transfer-Aware Runtime System for Heterogeneous Asynchronous Parallel Execution // ACM SIGARCH Computer Architecture News - HEART '15. 2015. Vol. 43. Issue 4. P. 40-45.

Methods for Computation Planning in the Parallel Dataflow Computing System "Buran"

D.N. Zmejev, A.V. Klimov, N.N. Levchenko, A.S. Okunev

Institute for Design Problems in Microelectronics of RAS, Moscow, zmejevdn@ippm.ru, klimov@ippm.ru, nick@ippm.ru, oku@ippm.ru

***Abstract* — The project of the parallel dataflow computing system (PDCS) "Buran" implements a new dataflow computing model with a dynamically formed context. This computing model has a number of features in comparison with the computing models used in traditional computing systems. The PDCS "Buran" is multi-core and scalable system. The computational core of the PDCS includes an execution unit, a matching processor, a token commutator, a hash block and other blocks. The content addressable memory of keys and token memory, which are part of the matching processor, have a limited capacity. This can cause memory overflow and, as a result, block the computational process. In order to ensure the solving of tasks in the PDCS, it is necessary to effectively manage local memory filling in the computational core. Computation planning allows the management of the computational process in automatic mode and the creation of possibility to solve large tasks, while having a limited capacity of content addressable memory. The article describes the planning methods associated with the distribution of computations by time. The first method spools and swaps tokens dynamically, using a special token-"lock" that sends all the tokens that are matched to it to the memory of spooled tokens. The programmer determines the ranges of spooled tokens by specifying the relevant key and the mask of the token-"lock". Simplicity of hardware support is an advantage of this method. The second method of computation planning uses spooling and swapping of active and passive stages. The group of tokens, which is a separate part of the program, is combined under a certain stage number. The task is divided into stages by the programmer using hash functions which generate the stage number by key. For the functioning of this method at the hardware level the followings are added: a block of hashing by time (generates the stage number), the stage table, the spooled memory, and the memory of the postponed tokens. This method is more versatile than the previous one, but it requires a lot of hardware and efforts from programmer to divide the task into stages. The third method is using only postponed stages. It is a development of the previous one, but with the refusal of the procedure of token spooling. This eliminates the potential of an overflow of the content addressable memory. Only tokens of active and postponed stages are involved in computation. The application of the methods described in the article makes it possible to solve tasks of large dimension, excluding the blocking of the computational process associated with overflow of the content addressable memory of keys. This**

work was carried out with the partial financial support of the RFBR grants 17-07-00478 и 17-07-00324.

***Keywords* — computation planning, parallel dataflow computing system, dividing into stages, dataflow computing model.**

REFERENCES

[1] Dongarra Dzh. Exascale future of supercomputers. Superkomp'jutery, 2010, no. 1(1), pp. 21–23 (in Russian).

[2] Sterling T. Etcetera of Sterling. Superkomp'juter, 2010, no. 3, pp. 17-20 (in Russian).

[3] Klimov A.V., Levchenko N.N., Okunev A.S. Advantages of dataflow computing model in a non-homogeneous networks. Zhurnal «Informacionnye tehnologii i vychislitel'nye sistemy», 2012, no. 2, pp. 36-45 (in Russian).

[4] Levchenko N.N., Okunev A.S., Stempkovskij A.L. The usage of dataflow computing model and architecture that implements it for exaflops performance system. Problemy razrabotki perspektivnyh mikro- i nanojelektronnyh sistem (MJeS), 2012, no.1, pp. 459-462 (in Russian).

[5] Stempkovskij A.L., Levchenko N.N., Okunev A.S., Cvetkov V.V. Parallel dataflow computing system - the further development of architecture and the structural organization of the computing system with automatic distribution of resources. Zhurnal «INFORMACIONNYE TEHNOLOGII», 2008, no. 10, pp. 2-7 (in Russian).

[6] Klimov A.V., Levchenko N.N., Okunev A.S., Stempkovskij A.L. The application and implementation issues of dataflow computing system. Problemy razrabotki perspektivnyh mikro- i nanojelektronnyh sistem (MJeS), 2016, no. 2, pp. 100-106 (in Russian).

[7] Zmeev D.N., Levchenko N.N., Okunev A.S., Stempkovskij A.L. The principles of organization of the I/O system of the parallel dataflow computing system. Programmnye sistemy: teorija i prilozhenija, 2015, no. 6:4(27), pp. 3–28 (in Russian). Available at: http://psta.psiras.ru/read/psta2015_4_3-28.pdf (accessed: 20.04.2018).

[8] Levchenko N.N., Okunev A.S., Jahontov D.E. Study of matching processor for the parallel dataflow computing system "Buran". Problemy razrabotki perspektivnyh mikro- i nanojelektronnyh sistem (MJeS), 2012, no. 1, pp. 467-470 (in Russian).

[9] Lee B., Hurson A.R. Dataflow Architectures and Multithreading. Computer, Aug 1994, vol. 27, no. 8, pp. 27-39.

[10] Silc J., Robic B., Ungerer T. Asynchrony in parallel computing: From dataflow to multithreading. Parallel and Distributed Computing Practices, 1998, vol. 1, no. 1, pp. 3-30.

[11] Soukaina N.Hmid, Jose G.F. Coutinho, and Wayne Luk. A Transfer-Aware Runtime System for Heterogeneous Asynchronous Parallel Execution. ACM SIGARCH Computer Architecture News - HEART '15, 2015, vol. 43, issue 4, pp. 40-45.