УДК 004.27

Средства распределения вычислений в ППВС «Буран» и варианты реализации блока выработки хэш-функций

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

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

Аннотация **– В статье описываются подходы к реализации возможности эффективного распределения вычислений как по вычислительным ядрам (по пространству), так и по этапам (во времени). Решение проблем с распределением вычислений в параллельной потоковой вычислительной системе позволяет максимально возможно загружать аппаратные ресурсы системы, резко уменьшать количество передач сообщений между ядрами и тем самым обеспечивать рост реальной производительности системы на широком круге задач. Описываются требования, предъявляемые к хэш-функциям распределения вычислений во времени и по пространству. Приводятся основные принципы создания различных функций распределения, а также варианты реализации и размещения блока вычисления хэш-функций в параллельной потоковой вычислительной системе.**

Ключевые слова **– локализация вычислений, параллельная потоковая вычислительная система, хэш-функции, варианты размещения БХФ.**

I. Введение

В традиционных системах, как известно, для большого количества актуальных задач с увеличением числа процессоров падает реальная производительность отдельного процессора. Проблема еще более усугубляется, когда в задаче присутствует активная работа с глобальными данными, причем для разных классов задач падение реальной производительности различно. Кроме того, в последнее время увеличивается число алгоритмов, которые эффективно распараллеливаются только до сотни ядер.

Также, потенциал многоядерных (до сотен тысяч и миллионов вычислительных ядер) вычислительных систем, а именно только такими и могут быть суперкомпьютеры, во всей полноте может использовать только параллельное программное обеспечение, которое обеспечивает распараллеливание вычислительных процессов. Необходимость параллельного программирования и трудности, которые с ним связаны, также являются большой проблемой для всех производителей компьютеров.

Вышеперечисленные проблемы, а также главный вопрос – эффективной загрузки аппаратных ресурсов высокопроизводительной системы, как отдельного кристалла, так и всей системы, предлагается решать посредством перехода к «новой» потоковой модели вычислений с динамически формируемым контекстом. Архитектура, реализующая такую модель вычислений, является предметом исследований авторов статьи.

Можно отметить, что в восьмидесятые годы прошлого столетия за рубежом велись работы над архитектурами вычислительных систем, реализующих модель вычислений dataflow [1-3], однако по разным причинам это направление работ было приостановлено.

В России эти исследования не «сворачивались». Одним из отличий разрабатываемой на базе потоковой модели вычислений с динамически формируемым контекстом параллельной потоковой вычислительной системы от предшествующих разработок является развитая система управления вычислениями, как в пространстве вычислительных ядер, так и во времени.

II. Архитектура параллельной потоковой вычислительной системы

Потоковая модель вычислений с динамически формируемым контекстом подробно рассмотрена в статьях [4, 5]. Базовая архитектура ППВС (рис. 1), реализующая эту модель вычислений, представляет собой многоядерную масштабируемую вычислительную систему [6, 7]. Помимо рассматриваемой в статье базовой архитектуры, авторами исследовались и другие варианты реализации [7]. Эти исследования выходят за рамки данной статьи.

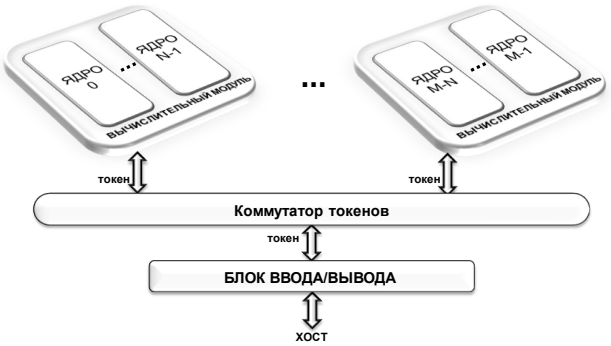

Рис. 1. Структурная схема ППВС

Основной источник взаимодействий в ППВС — это передача токенов (информационная структура данных,

в состав которой входит предаваемый операнд - данное), ключ с контекстом и адресом программного узла, маска, а также ряд других специальных признаков) между вычислительными ядрами. Коммутация между вычислительными ядрами осуществляется с использованием значения номера ядра.

Вычислительные ядра организуются в вычислительные модули (в рамках одного кристалла). Вычислительный модуль (рис. 2) системы конструктивно состоит из набора вычислительных ядер, в состав которых входят следующие основные блоки:

1) процессор сопоставления (ПС);
2) исполнительное устройство (ИУ);
3) внутренний коммутатор токенов;
4) внутренний коммутатор пакетов (пакет – это информационная структура, содержащая служебную информацию и данные).

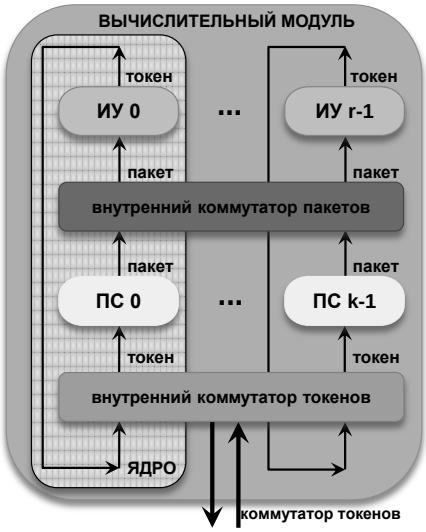

Рис. 2. Структурная схема вычислительного модуля ППВС

Процессор сопоставления является основным узлом вычислительного ядра ППВС «Буран», реализующим базовые принципы модели вычислений с управлением потоком данных и имеет свою систему команд. Процессор сопоставления содержит сопоставляющее устройство, память ключей которого реализуется в виде аппаратной ассоциативной памяти, разбитой на модули.

Процессор сопоставления в ППВС обеспечивает взаимодействие токенов различных типов в зависимости от содержимого поля «код операции» ключа токена, путем сравнения этого поля ключа и оценивая состояние других полей ключа токенов. Ключи токенов сопоставляются в ассоциативной памяти ключей, а сами токены хранятся в памяти токенов. В результате работы процессор сопоставления формирует пакеты, которые готовы к исполнению на «любом» свободном исполнительном устройстве.

Исполнительное устройство предназначено для обработки программных узлов потоковой программы. Программный узел может содержать как команды обработки поступивших в узел данных, так и команды изменения контекста программного узла. Контекст определяет положение данного в виртуальном адресном пространстве задачи. Следовательно, в программе узла можно выделить команды обработки данных и команды обработки контекста. В основном эти команды не имеют взаимной зависимости по данным и, следовательно, могут быть выполнены параллельно.

Внутренний коммутатор пакетов необходим для распределения готовых пакетов между свободными исполнительными устройствами в рамках одного вычислительного модуля. Коммутатор пакетов обеспечивает равномерную загрузку всех исполнительных устройств за счет направления готовых пакетов в наименее заполненное ИУ, обеспечивая тем самым связь всех процессоров сопоставления со всеми исполнительными устройствами внутри одного вычислительного модуля. Использование подобного коммутатора способствует сглаживанию имеющейся неоднородности параллелизма задачи, поскольку каждый процессор сопоставления генерирует готовые к выполнению пакеты неравномерно. Коммутатор пакетов перекладывает распределение имеющихся аппаратных ресурсов с программиста на аппаратуру, фактически являясь аппаратным балансёром.

Внутренний коммутатор токенов распределяет токены по процессорам сопоставления на основе номера вычислительного ядра, который хранится в одном из полей токена.

Архитектура ППВС масштабируема и при увеличении числа ядер в системе падение реальной производительности на задачах со сложно организованными данными происходит существенно медленнее, чем при решении подобных задач на вычислительных системах с классической архитектурой. Это достигается, во-первых, за счет использования самого принципа потока данных, когда готовые к выполнению данные активируют выполнение программы узла; при этом, активируемому узлу для своего полного выполнения не требуется никаких дополнительных данных; во-вторых, узлы выполняются полностью независимо друг от друга, и, в-третьих, благодаря правильному выбору хэш-функции (функции распределения вычислений по группам ядер), сокращающей число межъядерных передач токенов.

Используя такую модель вычислений, реализуемую в архитектуре ППВС, создание и выполнение программы не зависят от конкретной конфигурации вычислительной системы, то есть программа будет работать и на одном ядре, и на любом другом числе

вычислительных ядер без повторной компиляции. Для улучшения прохождения программы на различных конфигурациях требуется только изменить параметры или саму хэш-функцию.

III. Применение функций распределения в ППВС

Выбор способа распределения виртуальных узлов по вычислительным ядрам для каждой конкретной задачи является одним из определяющих факторов повышения эффективности вычислений ППВС. Распределение вычислений в ППВС применяется прежде всего для обеспечения равномерной загруженности работой вычислительных ядер системы, затем для минимизации нагрузки на коммуникационную сеть, а также для уменьшения требуемого объема ассоциативной памяти ключей. Равномерность нагрузки на вычислительные ядра и на коммуникационную сеть часто вступают в противоречие друг с другом, поскольку улучшение по одному показателю приводит к ухудшению по другому. Поэтому при выборе характера распределения вычислений приходится учитывать влияние распределения на оба показателя. Данные показатели существенно зависят от распределения виртуальных узлов по ядрам ППВС.

A. Распределение вычислений во времени

Распределение вычислений (другими словами, локализация вычислений) во времени с помощью задаваемых пользователем хеш-функций является универсальным методом планирования вычислений в ППВС. Все токены вычислительного процесса разделяются с помощью хэш-функций на некоторое количество групп, которые будем называть временными этапами (или просто этапами). Все этапы делятся на «активные» и «пассивные».

Токены «активных» этапов заполняют память процессора сопоставления, где происходит сопоставление токенов, принадлежащих соответствующему этапу. Токены «пассивных» этапов находятся в памяти, где сопоставлений нет. Чтобы «пассивный» этап мог продолжить работу, он должен сначала стать «активным».

В отличие от классических систем в ППВС в определенные моменты времени некоторый «активный» этап может стать «пассивным», то есть происходит «деактивация» «активного» этапа, а некоторый «пассивный» этап может стать «активным». Событие первого типа называется откачкой, второго – подкачкой этапа.

Работа с этапами осуществляется через таблицу «активных» этапов, куда заносятся номера всех этапов (незавершенных) с пометками об активности. По завершении работы этапа (когда отсутствуют токены принадлежащие данному этапу и ожидающие сопоставление с другими токенами) этап исключается из таблицы этапов, тем самым переходя в разряд «пассивных».

B. Распределение вычислений по пространству

При формировании токены проходят процедуру вычисления хэш-функции, которая предварительно задается программистом. На основе целевого адреса (расширенного контекста токена, который определяет его положение в виртуальном адресном пространстве задачи) вычисляется номер вычислительного ядра, куда должен поступить данный токен. Фактически это и есть распределение вычислений по пространству. Если получившийся номер отличается от номеров вычислительных ядер, входящих в вычислительный модуль, в котором сгенерирован токен, то токен отправляется в коммуникационную сеть. В противном случае он поступает через внутренний коммутатор токенов в требуемое вычислительное ядро.

C. Требования предъявляемые к функциям распределения

Функция распределения должна зависеть только от полей контекста и может быть выражена в виде некоторой формулы, параметрами которой являются поля контекста. Основными операциями таких формул для обеспечения максимального быстродействия должны быть побитовые логические операции, операции сдвига (умножение или деление на степень двойки), и некоторые специальные базисные функции. Иногда могут применяться также операции целочисленного сложения или вычитания. Для получения окончательного значения номера вычислительного ядра берется остаток от деления результата на число ядер N конкретной конфигурации вычислительной системы.

Для работы описанного механизма в любом случае от программиста требуется хорошая (оптимальная) функция распределения. При ее выборе или создании такой функции следует руководствоваться следующими основными принципами:

1) функция распределения должна быть вычислительно простой;

2) при распределении во времени, токены, порождаемые некоторым узлом, должны принадлежать этапу, номер которого равен или больше номера этапа, порождающего эти токены;

3) при распределении во времени, каждый этап должен с запасом размещаться в «активной» зоне (ассоциативная память ключей), желательно даже, чтобы несколько соседних этапов могли помещаться в «активной» зоне;

4) при распределении по пространству, функция распределения должна обеспечивать наилучшую равномерность распределения на всех уровнях коммуникационной иерархии (в предположении, что иерархическая близость соответствует арифметической, то есть вычислительные ядра с близкими номерами близки и в коммуникационной сети).

D. *Примеры выбора хэш-функций для различных задач*

Как было отмечено в работе [5] был создан ряд хэш-функций (ХФ), которые значительно повышают эффективность вычислений для определенных типов приложений, они подробно описаны в статье [8]:

1) хэш-функция БПФ создана специально для выполнения задачи быстрого преобразования Фурье [9], она позволяет минимизировать число пересылок в «чужие» ядра токенов в процессе вычислений;

2) хэш-функция ZIP, принимает два двоичных аргумента и скрещивает их разряды, что оказывается полезны для задач, где требуются многомерные (2 и более измерений) решетки с доступом к ближайшим соседям, а также для операций над плотно заполненными матрицами;

3) хэш-функция NORM, которая «нормализует» первый аргумент n и затем берет k старших разрядов результата (под нормализацией понимается сдвиг аргумента влево так, чтобы его старшая единица вышла за пределы разрядной сетки числа), полезна в алгоритмах свертки по методу сдваивания (вычисление суммы массива, минимума, максимума и подобных ассоциативных операций);

4) хэш-функция BLK, обеспечивает наилучшую равномерность распределения однородного массива элементов с индексами i от 0 до N-1 на всех уровнях коммуникационной иерархии (в предположении, что модули с близкими номерами близки коммуникационно);

5) хэш-функция STD, стандартная функция равномерно распределяет данные по вычислительным ядрам системы;

6) хэш-функция FLD, обеспечивает распределение данных равными порциями по вычислительным ядрам одномерного массива элементов;

7) хэш-функция HMerge, может найти применение в алгоритмах с тесной связностью, где требуется комбинировать попарно элементы большого множества, но есть способ сделать это быстрее, чем перебирая N^2 комбинаций.

IV. Варианты размещения блока выработки хэш-функций в ППВС

В ППВС за реализацию работы с функциями распределения отвечает блок вычисления хэш-функций (рис. 3). Поскольку применяемые функции для распределения вычислений по пространству и во времени схожи, то имеет смысл сделать единым блок, который будет работать как с тем, так и с другим распределением. При невозможности размещения в одном месте (узле), он просто будет физически дублироваться.

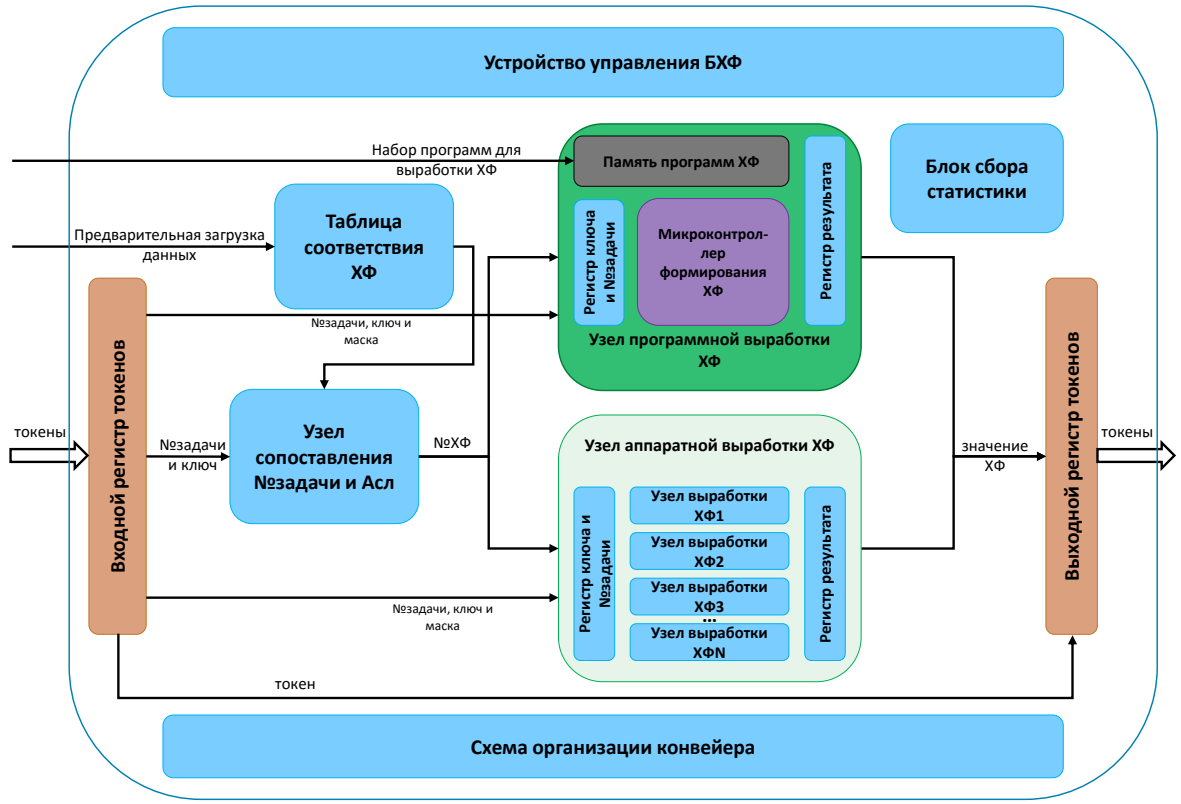

Рис. 3. Функциональная схема блока вычисления хэш-функций

Кратко опишем работу блока. Перед стартом задачи, происходит загрузка требуемыми значениями блоков «Таблица соответствия ХФ» и «Память программ ХФ» узла программной выработки хэш-функции. В «Таблицу соответствия ХФ» записываются номера задач, подзадач, номера программных узлов и

соответствующие им номера аппаратных и программных хэш-функций. В «Память программ ХФ» помещаются микропрограммы выработки хэш-функций, которые предварительно создает разработчик для своей программы, если базовый, реализованный аппаратно, набор хэш-функций его не удовлетворяет.

Работа блока хэш-функций начинается с прихода токена на «входной регистр токенов». Поля «№ задачи» и «Ключ» передаются в узел сопоставления, в котором происходит определение номера хэш-функций (№ХФ) для данного токена. Согласно этому номеру хэш-функций поля «№задачи», «ключ» и «маска» поступают либо в «Узел программной выработки ХФ», либо в «Узел аппаратной выработки ХФ».

Если токен использует аппаратную хэш-функций, то в «Узле аппаратной выработки ХФ» по номеру выбирается хэш-функций, которая преобразует исходные данные в значение, соответствующее номеру вычислительного ядра или номеру этапа в программе.

В случае использования программной хэш-функций, из «Памяти программ ХФ» выбирается соответствующая программа, по которой микроконтроллер формирует значение, соответствующее номеру вычислительного ядра или номеру этапа в программе.

Результирующее значение записывается в «Выходной регистр токенов» в поле, которое в зависимости от функции, выполняемой блоком выработки хэш-функции, может быть, как «номером этапа» при работе внутри ядра, так и «номером вычислительного ядра» при пересылке токена между ядрами.

Работа блока выработки хэш-функции конвейеризирована для обеспечения высокого темпа приема и выдачи токенов.

Блок выработки хэш-функций (БХФ) позволит определять конкретную функцию распределения, которая задается пользователем вместе с программой, как для всей задачи, так и для каждого программного узла в отдельности.

Как уже упоминалось, процедуре вычисления хэш-функции подвергается ключ каждого токена, сформированного в вычислительной системе или пришедшего в неё извне. Поэтому БХФ обязательно должен быть реализован в «интеллектуальном» устройстве ввода/вывода данных. Блок ввода/вывода ППВС предназначен для преобразования поступающих на его вход «обычных» данных в токены, выработки хэш-функций, а также обратного преобразования выдаваемых на хост токенов в данные. Все входные данные, поступающие в систему извне, должны пройти процедуру хэширования для того, чтобы определить номер вычислительного ядра, в который они должны будут поступить. Затем происходит анализ на основе номера этапа токена – к «активным» или «пассивным» этапам относится этот токен, то есть следует ли его задержать в блоке ввода/вывода до того момента, пока этап этого токена станет «активным». Фактически тем самым определяется очередность поступления токенов в коммутатор токенов вычислительной системы, и, таким образом, происходит распределение вычислений во времени при котором в коммуникационную сеть системы не вводятся токены «пассивных» этапов.

Также БХФ может быть размещен непосредственно в процессоре сопоставления, на его входе. При реализации в процессоре сопоставления все приходящие токены проходят через процедуру хэширования для выработки номера этапа. На основании этого номера определяется, будет ли продолжена работа над данным токеном – в случае если он относится к «активному» этапу, или он будет приостановлен – в случае если он относится к «пассивному» этапу.

Кроме распределения во времени, в процессоре сопоставления может выполняться распределение по пространству. Это происходит в том случае, когда в результате сопоставления токенов вместо пакета образуется новый токен – при взаимодействии специальных токенов (таких как «токен-счетчик», «токен-сумма» и другие), что требует размещения БХФ на выходе процессора сопоставления.

И, наконец, БХФ может быть размещен на выходе исполнительного устройства. После формирования нового токена в исполнительном устройстве требуется определить номер вычислительного ядра, в который он должен будет поступить. Функциональность БХФ может быть реализована в самом исполнительном устройстве, путем выполнения программы, находящейся в памяти программ. Но поскольку вычисление самой функции распределения может выполняться параллельно с подготовкой других полей токена, и только после формирования полей расширенного контекста токена, то эта программа должна быть внедрена непосредственно в программный узел, что противоречит постулату о независимости программы от функций распределения. Или в противном случае потребуется существенное усложнение самого исполнительного устройства. При этом, все равно не будет решена проблема вычисления хэш-функции на входе в процессор сопоставления. Поэтому более правильным и является размещение отдельного блока выработки хэш-функции сразу после исполнительного устройства. К преимуществам данного варианта следует отнести отказ от вычисления хэш-функции на выходе процессора сопоставления, поскольку сформировавшийся в нем токен в обход ИУ может поступать на этот блок.

Реализация БХФ непосредственно на исполнительном устройстве будет эффективной только в том случае, если ИУ будет совмещено с процессором сопоставления (фактически, Processor in Memory). В этом случае архитектура вычислительной системы

претерпевает ряд незначительных изменений, одним из которых является отказ от внутреннего коммутатора пакетов. В то же время, все вычисления функции распределения (помимо блока ввода данных) будут сосредоточены внутри процессора сопоставления.

Также имеется дополнительная возможность программной реализации БХФ на хост-машине, что позволит запускать в ППВС только токены, принадлежащие к «активным» этапам.

V. Выводы

Выбор базисных функций может определяться ориентацией системы на конкретный класс задач. Для таких функций характерно, что они достаточно сложно реализуются программно в традиционной системе команд, что является доводом для их специальной реализации в аппаратуре. Такая реализация возможна либо в форме дополнительных специальных команд, либо в составе отдельного таблично-программируемого блока.

Вариант реализации (аппаратный или программный) блока вычислений хэш-функций следует выбирать прежде всего исходя из планируемого размещения блока в узлах вычислительной системы. Аппаратная реализация возможна либо в форме заранее определенного набора хэш-функций, либо с использованием программируемой логической интегральной схемы, в которую пользователь сможет загружать наиболее эффективные для решения своих задач хэш-функции. Вариант реализации определенного набора хэш-функций оптимален по занимаемой площади, а вариант с ПЛИС предоставляет универсальную возможность использования хэш-функций.

Эксперименты показали, что при использовании хорошо подобранной функции распределения и правильного размещения БХФ общее время параллельного выполнения программы значительно сокращается за счет улучшения равномерности распределения и существенного сокращения объема передаваемой информации на всех уровнях иерархии вычислительной системы.

Литература

[1] Lee Ben, Hurson A.R. Issues in Dataflow Computing //Advances in computers. 1993. Vol.37. P. 285-333.

[2] Ben Lee, Hurson A. R. Dataflow Architectures and Multithreading//Computer. 1994. Aug. V. 27, no. 8. P. 27-39.

[3] 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.

[4] 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.

[5] Климов А. В., Левченко Н. Н., Окунев А. С., Стемпковский А. Л. Суперкомпьютеры, иерархия памяти и потоковая модель вычислений // Программные системы: теория и приложения: электрон. научн. журн. 2014, Т. 5, № 1(19), С. 15-36, URL: http://psta.psiras.ru/read/psta2014_1_15-36.pdf (дата обращения: 30.03.2016)

[6] 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 // Всероссийская научно-техническая конференция «Проблемы разработки перспективных микро- и наноэлектронных систем – 2012» (МЭС-2012): сборник трудов. – М.: ИППМ РАН, 2012, С. 459-462.

[7] 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.

[8] Стемпковский А.Л., Климов А.В., Левченко Н.Н., Окунев А.С. Методы адаптации параллельной потоковой вычислительной системы под задачи отдельных классов // журнал «Информационные технологии и вычислительные системы», 2009 г. № 3, С. 12-21.

[9] Левченко Н.Н., Окунев А.С. Специализация архитектуры многоядерной параллельной потоковой вычислительной системы для решения задачи быстрого преобразования Фурье // Сб. науч. трудов IV Всероссийской Научно-технической конференции «Проблемы разработки перспективных микро- и наноэлектронных систем 2010». М.: ИППМ РАН. 2010. С. 458-461.

The Tools for Computation Distribution in the PDCS "Buran" and Hash-Functions Block Implementation Options

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

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

***Keywords* — localization of computation, parallel dataflow computing system, BHF implementation options.**

ABSTRACT

Problems of computing on high-performance systems with a large number of cores can be solved by moving to the dataflow computing model with dynamically formed context. The key issues - the effective loading of hardware resources of the computing system, as well as the possibility of computation scheduling by time.

This article provides the general description of the PDCS "Buran" architecture. It describes the structure of the computing module and the computing core of the PDCS. Authors declare that the choice of the program nodes among the computing cores distribution method directly defines the growth of execution effectiveness of this task on the PDCS.

The requirements for functions of the computation distribution in time and by space are described. The "Active" and "passive" phases are defined, the method of core number computation is described. The basic principles for programmer to choose the propriety hash-functions of distribution are presented. Various placing options of the hash-functions block in the PDCS are presented. The reasons for placement the hash-functions block in various nodes and blocks of the PDCS, such as the I/O block, the execution unit and the matching processor, are given. Both hardware and software implementation options of the hash-functions block allocation among the computing cores (by space) and for the gradation of the task (in time) are described.

REFERENCES

[1] Lee Ben, Hurson A.R. Issues in Dataflow Computing //Advances in computers. 1993. Vol.37. P. 285-333.

[2] Ben Lee, Hurson A. R. Dataflow Architectures and Multithreading//Computer. 1994. Aug. V. 27, no. 8. P. 27-39.

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

[4] Klimov A.V., Levchenko N.N., Okunev A.S. The Advantages of Dataflow Calculation Model in Heterogeneous Networks Conditions. Zhurnal «Informacionnye tehnologii i vychislitel'nye sistemy», 2012, No. 2, pp. 36-45 (in Russian).

[5] Klimov A.V., Levchenko N.N., Okunev A.S., Stempkovskij A.L. Supercomputers, memory hierarchy and the dataflow computing system. Programmnye sistemy: teoriya i prilozheniya: elektron. nauchn. zhurn. 2014, vol. 5, No. 1(19), pp. 15-36 (in Russian). Available at: http://psta.psiras.ru/read/psta2014_1_15-36.pdf (accessed 30.03.2016).

[6] Levchenko N.N., Okunev A.S., Stempkovskij A.L. Dataflow Calculation Model and Its Architecture Usage for Exaflop/s Computing Systems. Vserossijskaja nauchno-tehnicheskaja konferencija «Problemy razrabotki perspektivnyh mikro- i nanojelektronnyh sistem – 2012» (MJeS-2012): sbornik trudov. IPPM RAN, 2012, pp. 459-462 (in Russian).

[7] Stempkovskij A.L., Levchenko N.N., Okunev A.S., Cvetkov V.V. Parallel Dataflow Computing System: Further Development of Architecture and Structural Organization of the Computing System with Automatic Distribution of Resources. Zhurnal «INFORMACIONNYE TEHNOLOGII». 2008, No. 10, pp. 2-7 (In Russian).

[8] Stempkovskij A.L., Klimov A.V., Levchenko N.N., Okunev A.S. Methods of Parallel Dataflow Computing System Adaptation for Problems of Individual Classes. zhurnal «Informacionnye tehnologii i vychislitel'nye sistemy». 2009, No. 3, pp. 12-21 (In Russian).

[9] Levchenko N.N., Okunev A.S. Specialized the architecture of the parallel multicore dataflow computing system for solution of task FFT. Sb. nauch. trudov IV Vserossijskoj Nauchno-tehnicheskoj konferencii «Problemy razrabotki perspektivnyh mikro- i nanojelektronnyh sistem 2010». IPPM RAN, 2010, pp. 458-461 (in Russian).