УДК 004.27 DOI 10.31114/2078-7707-2018-3-24-30

Преимущества потоковой модели вычислений

Н.Н. Левченко, А.С. Окунев, А.Л. Стемпковский

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

***Аннотация* — Одной из основных задач в области высокопроизводительных вычислений является решение проблемы эффективного распараллеливания вычислений. В данной статье описывается архитектура вычислительной системы, реализующая оригинальную потоковую модель вычислений с динамически формируемым контекстом. Представлены основные принципы работы этой модели вычислений. Проведен анализ особенностей и преимуществ потоковой модели вычислений. Сделаны выводы о перспективности использования потоковой модели вычислений для создания в будущем новых высокопроизводительных вычислительных систем.**

***Ключевые слова* — потоковая модель вычислений, архитектура, перспективные вычислительные системы, масштабирование вычислений.**

I. Введение

Практически с начала 21 века в области создания микропроцессоров наметилась тенденция к увеличению числа вычислительных ядер в рамках одного кристалла. Особенно это проявилось при создании графических ускорителей, в которых на один кристалл приходится более 5000 простых процессоров (stream processors) [1]. Эта тенденция связана с тем, что, во-первых, производительность одного вычислительного ядра на используемой элементной базе практически достигла своего предела из-за необходимости отвода тепла с единицы площади кристалла, а во-вторых, число транзисторов, которые можно разместить на кристалле, продолжает увеличиваться.

В одном из докладов [2] приводятся основные тенденции развития в области высокопроизводительных вычислений. Стоит обратить внимание на такие показатели как число узлов (от 100 тыс. до 1 миллиона), а также общий уровень параллельности - 10^9. Эти показатели должны быть достигнуты к 2022 году.

Всё это демонстрирует основную проблему в области высокопроизводительных вычислений – проблему эффективного и достаточного простого метода распараллеливания вычислений. В традиционных вычислительных системах кластерного типа данная проблема решается в основном на программном уровне применением параллельных алгоритмов и оптимизирующих компиляторов. На аппаратном уровне распараллеливание производится на уровне операций в окне из ограниченного числа команд [3].

В последние годы зарубежные страны вкладывают огромные средства в разработку и создание вычислительных систем эксафлопсного уровня производительности, в том числе и с использованием новых моделей вычислений. К примеру, США тратят на научно-исследовательские и опытно-конструкторские разработки в этой области около 1-2 млрд долларов в год. Китай – более 1 млрд долларов в год выделяет на проведение аналогичных исследований. В Японии запланированы инвестиции более 1 млрд долларов на 5 лет. Евросоюз всего потратит на подобные работы около 5 млрд евро [4]. Причем финансирование этих программ в ведущих зарубежных странах ведется в основном государством с привлечением частного капитала. В Японии, например, создается процессор новой архитектуры, реализующий потоковую модель вычислений (dataflow) и новое программное окружение [5]. США, Китай и Евросоюз планируют внедрять в различные отрасли не одну систему эксафлопсной и трансэксафлопсной производительности, а множество подобных систем. Предложения же отечественных разработчиков вычислительных систем высокой производительности, в основном, не выходят за рамки классических кластерных подходов [6]-[7].

Архитектура параллельной потоковой вычислительной системы (ППВС), реализующая оригинальную потоковую модель вычислений с динамически формируемым контекстом [8], разрабатываемая в ИППМ РАН и позволяющая эффективно распараллеливать вычисления, как раз и является перспективной для применения в будущих высокопроизводительных системах.

Особенности архитектуры ППВС позволят приблизиться к решению проблем создания (таких как, энергопотребление, отказоустойчивость, масштабируемость и др.) эксафлопсных систем.

II. Потоковая модель вычислений и архитектура ППВС

В основе потоковой модели вычислений с динамически формируемым контекстом лежат следующие основные понятия:

- токен;
- парадигма «раздачи»;
- активация вычислений по готовности данных.

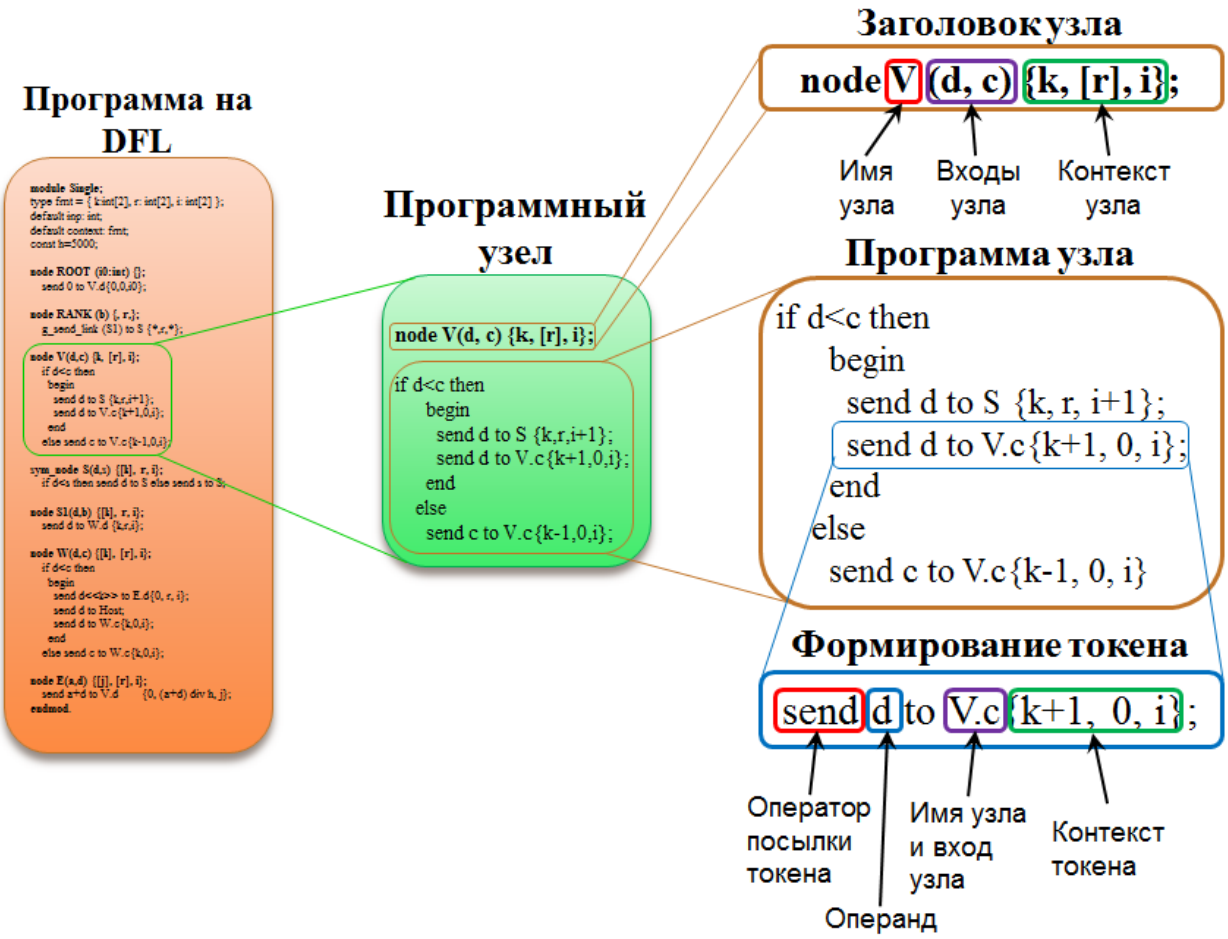

Рис. 1. Структура программы на параллельном языке DFL ППВС

В модели вычислений «циркулируют» исключительно токены. Токены представляют собой структуру данных, которая содержит операнд, ключ (тег), однозначно определяющий положение операнда в виртуальном адресном пространстве задачи, а также набор служебных полей, поддерживающих взаимодействие токенов между собой.

Потоковая модель вычислений с динамически формируемым контекстом воплощается в параллельном языке программирования DFL [9]. Программа на языке DFL представляет собой набор программных узлов (рис. 1). Каждый из программных узлов состоит из заголовка узла и программы узла. Заголовок узла состоит из служебного слова, определяющего тип программного узла (на рис. 1 стандартный узел «node»), имени узла, набора входов узла (в предыдущих версиях языка существовало ограничение на число входов, равное двум) с описанием их типа, а также контекста (тега или индекса) программного узла, в рамках которого обрабатываются поступившие на вход узла операнды. Контекст, как и в традиционной динамической модели dataflow [10]-[12], обеспечивает многократную активацию одного программного узла для операндов различных виртуальных адресов пространства задачи. Активация программного узла происходит только в том случае, если все необходимые для его выполнения данные поступили на его входы. Другими словами, если узел содержит 5 входов, то для его активации требуется приход пяти токенов, причем контексты (индексы) с маской и адреса программного узла пришедших токенов должны совпадать.

Использование основополагающих принципов ассоциативной памяти наиболее эффективным образом реализует базовый принцип управления потоком данных, заключающийся в активации по готовности данных.

После активации программного узла происходит выполнение последовательности команд, в результате чего вычисляются новые данные. Причем для выполнения программного узла используются только начальные данные, элементы контекста и константы, то есть в потоковой модели вычислений отсутствует само понятие приостановки выполнения программного узла на подкачку необходимых данных. Результаты выполнения программного узла в виде токенов отсылаются либо в другие программные узлы, либо выдаются на ХОСТ-машину. При формировании токенов используется служебное слово «send». Помимо операнда в этом операторе посылки токена указывается имя программного узла и вход, на который будет

направлен токен, а также контекст токена, вычисляемый в коде программного узла.

Всякое значение, необходимое для выполнения программного узла, должно быть отправлено другим программным узлом на один из входов. Когда на часть входов узла данные уже поступили, то до прихода остальных данных, они должны где-то находиться в ожидании. Для этого ожидания и используется ассоциативная память. Для глобальной ассоциативной памяти системы «адресом» является ключ токена, в состав которого входит номер узла с набором индексов.

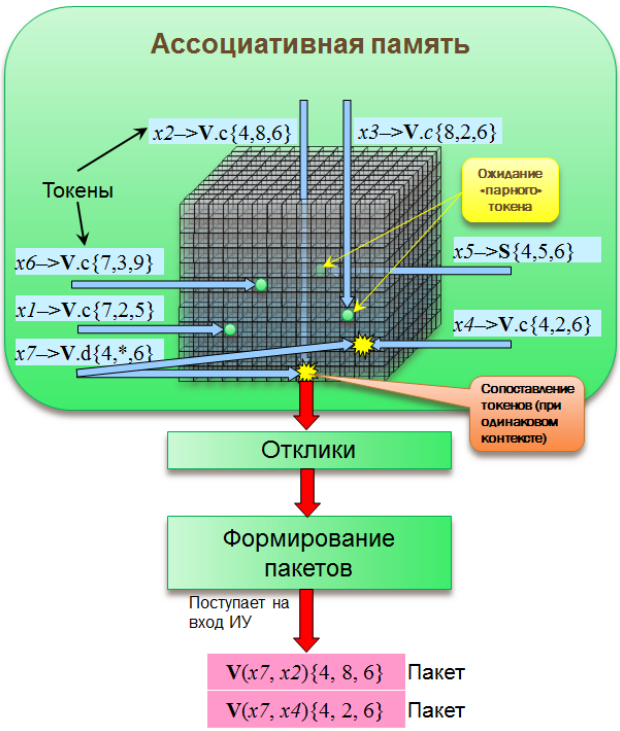

Рис. 2. Принцип работы модели вычислений ППВС

На рис. 2 изображен базовый принцип работы потоковой модели вычислений. Для простоты будем считать, что имеется одно глобальное устройство, выполняющее базовый принцип модели вычислений - ассоциативный поиск. В нашем случае таким устройством является ассоциативная память. Токены поступают на вход ассоциативной памяти. Далее происходит сопоставление токена со всеми имеющимися в ассоциативной памяти токенами. Токены могут остаться в ассоциативной памяти в двух случаях: во-первых, когда пришедший токен не обнаружил в ассоциативной памяти «парного» ему, а во-вторых, если в процессе сопоставления его кратность оказалась отличной от нулевой. Кратность обеспечивает возможность токену участвовать в сопоставлении со множеством токенов (при каждом «удачном» сопоставлении, кратность уменьшается на единицу) и только при исчерпании кратности токена он удаляется. В ППВС механизм кратности работает так, что при взаимодействии один из токенов в обязательном порядке будет удаляться. В случае «удачного» сопоставления происходит формирование пакетов, готовых к активации (то есть обработке в соответствующем программном узле) данных. «Удачным» является такое сопоставление, в результате которого все поля ключа токенов совпадают с учетом маскирования (на рис. 2 поле с маской обозначено символом «*»). Маскирование обеспечивает принудительное совпадение полей.

Архитектура вычислительной системы представляет собой набор вычислительных модулей соединенных коммуникационной сетью. Обращение к вычислительной системе происходит через блок ввода/вывода, который также соединен с коммуникационной сетью.

Вычислительный модуль, в свою очередь, состоит из некоторого числа вычислительных ядер, каждое из которых включает внутренний коммутатор токенов, процессор сопоставления (ПС) [13], коммутатор пакетов и исполнительное устройство (ИУ). Входные данные поступают на блок ввода/вывода, где они преобразуются в токены и проходят процедуру хэширования, в результате которой получают номер вычислительного ядра (ВЯ). После этого они подаются в коммуникационную сеть, которая доставляет их до требуемого ВЯ (в соответствии с полученным номером). Поступая в ВЯ токен приходит в ПС, в состав которого входят ассоциативная память ключей (АПК) и прямоадресуемая память токенов (ПТ). Сначала происходит сопоставление ключа токена с ключами других (ранее пришедших) токенов в АПК, которая содержит ключи и адреса токенов, находящихся в ПТ. В состав ключа токена входят поля, определяющие номер программного узла и контекст (при аппаратной реализации ключ составляет 64 разряда).

В случае совпадения с ключами других токенов происходит проверка – все ли требуемые для активации программного узла токены присутствуют. Если это так, то происходит формирование пакета, и он через коммутатор пакетов поступает на обработку на любое свободное ИУ вычислительного модуля. Обычно на каждый вход экземпляра программного узла приходит по одному токену. После формирования пакета токены и ключи удаляются из ПС. Однако этому может воспрепятствовать механизм кратности. В памяти программ каждого ИУ хранятся копии программ и констант, которые загружаются из хост-процессора перед началом работы. При отсутствии всех требуемых для активации программного узла токенов ключ токена занимает свободное место в АПК, а сам токен записывается в ПТ в ожидании поступления требуемых для активации токенов.

Токены с общим ключом (ключ и накладываемая на него маска) оказываются в одном и том же ПС благодаря функции хэширования (распределения вычислений), которая зависит только от ключа токена. В результате обеспечивается эффективная и распределенная организация ассоциативного поиска. Созданы различные функции распределения. Одни обеспечивают хорошее перемешивание данных, но нарушают имеющуюся локальность вычислений, другие – нацелены на сохранение локальности и, тем самым, обеспечивают минимизацию обращений к коммуникационной сети,

что повышает эффективность работы всей системы. Пользователь может либо выбрать хэш-функцию из имеющихся и изменить параметры, либо определить свою собственную хэш-функцию, разработанную под конкретную задачу.

Токены, сформированные в результате обработки пакета в ИУ, также подвергаются процедуре хэширования, после чего номер вычисленного ВЯ сравнивается с номером текущего, и при совпадении номеров токен направляется непосредственно на свой ПС, минуя глобальный коммутатор токенов.

III. Перспективность потоковой модели вычислений с динамически формируемым контекстом для будущих высокопроизводительных систем

Как уже говорилось, основным вызовом, стоящим перед перспективными высокопроизводительными вычислительными системами, является эффективная система создания и масштабирования параллельных программ для выполнения на сотнях тысяч вычислительных ядер. Для архитектуры ППВС «Буран» [14]-[15] с этой точки зрения характерно следующее:

A. Простота создания параллельных программ

Это обеспечивается, во-первых, тем, что программу для вычислительной системы на архитектуре ППВС можно создавать и отлаживать на одном вычислительном ядре. Отлаженная программа будет автоматически распараллелена на любое число ядер (текст программы не меняется и остается одним и тем же как для одного ядра, так и для сотен тысяч ядер). Во-вторых, программист имеет возможность менять распределение данных по вычислительным ядрам отдельно от текста программы. Для этого он может предложить свою функцию распределения вычислений или выбрать одну из имеющихся функций и с помощью изменения параметров настроить ее под специфику своей программы. Программист, таким образом, не занимается распараллеливанием программы, это делает аппаратура.

B. Аппаратное управление памятью

В отличие от традиционных систем в ППВС для программиста «скрыта» работа с памятью на уровне взаимодействия программных узлов между собой, то есть на уровне распределения вычислений по вычислительным ядрам. Помимо этого в ППВС отсутствует как таковая проблема когерентности кэш-памяти.

Работа с памятью доступна только на локальном уровне, внутри программного узла.

C. Аппаратное экстрагирование неявного параллелизма

Архитектура параллельной потоковой вычислительной системы позволяет в динамике (в процессе вычислений) выявлять неявный для программиста параллелизм задачи, который изначально заложен в самой программе. Например, такой параллелизм может быть выявлен на стыке разных итераций (при условии отсутствия обязательной синхронизации между ними), между разными активациями программных узлов, а также между разными программными узлами.

D. Парадигма «раздачи»

Программа для потоковой модели вычислений создается в парадигме «раздачи». Для этой парадигмы характерно, что каждый «производитель» нового данного имеет информацию о том, где будет оно востребовано, и поэтому он обеспечивает самостоятельную его рассылку по требуемым адресам (рис. 3). «Получатель» в этом случае просто ожидает прихода данных, и ему не нужна информация об их источнике. Применение парадигмы «раздачи» обеспечивает более эффективную стратегию перемещения данных, поскольку на каждое использование операнда приходится лишь одно сообщение.

Рис. 3. Парадигмы «сбора» (а) и «раздачи» (б)

В используемой традиционно парадигме «сбора», в свою очередь, только «потребитель» данных имеет информацию о том, какие данные ему требуются, и где их можно «взять», соответственно, «потребитель» их запрашивает, обращаясь к переменной или элементу массива. Поэтому аппаратуре трудно «понять», когда и где то или иное данное потребуется. Таким образом на одно данное в парадигме «сбора» приходится три сообщения – одно на запись и по два на каждое из считываний (запрос на получение операнда и передача операнда).

E. Аппаратное распределение вычислительного процесса по вычислительным ядрам

Распределение вычислений в ППВС происходит посредством хэш-функций, которые могут быть реализованы как аппаратно, так и задаваться программно. Хэш-функция вырабатывается только на основе ключа токена и обеспечивает распределение токенов по вычислительным ядрам, а также гарантирует, что токены, которые должны сопоставиться, будут направлены в один процессор сопоставления. С помощью хэш-функций программист может настроить как степень параллельности выполнения задачи, так и степень ее локальности, сократив тем самым объем передаваемых данных по коммуникационной сети вычислительной системы.

F. Высокая степень масштабируемости задач

В архитектуре ППВС падение реальной производительности при увеличении числа вычислительных ядер происходит существенно медленнее, чем при решении тех же задач на традиционных системах. Особенно это заметно на задачах со сложноорганизованными или многомасштабными данными. Этому способствует:

- сам принцип потока данных (готовые к выполнению данные активируют выполнение программы узла);
- локальность программного узла (активируемому узлу для своего полного выполнения не требуется никаких дополнительных данных), то есть обрабатываемый программный узел не прерывается на подкачку дополнительных данных;
- независимость выполнения программного узла (от других программных узлов);
- один и тот же протокол взаимодействия (между вычислительными модулями, представляющими собой группу ядер, и внутри их действует единый протокол, осуществляющий доставку токенов к ядрам).

Высокая масштабируемость задач является одним из главных преимуществ потоковой модели вычислений и реализующей ее архитектуры. К тому же выполнение параллельной программы и ее создание не зависят от конкретной конфигурации вычислительной системы.

G. Низкий семантический разрыв

Современные компьютеры имеют большой разрыв между операциями, описываемыми в языке программирования, и соответствующими операциями, реализуемыми аппаратурой вычислительной системы (его называют семантическим разрывом). Фактически есть семантический разрыв между архитектурой системы и средой программирования. Все это порождает большие проблемы – увеличивается стоимость разработки программного обеспечения, падает его надежность, усложняется операционная система и разрабатываемые компиляторы.

Семантический разрыв между параллельным языком высокого уровня (DFL) и архитектурой высокопроизводительной вычислительной системы «Буран» существенно ниже, чем в вычислительных системах, базирующихся на традиционных принципах.

Таким образом, в архитектуре ППВС обеспечивается рост надежности программного обеспечения, продуктивности программирования, эффективности использования аппаратуры вычислительной системы, а также сокращение времени решения задачи и снижение стоимости создания программного обеспечения.

H. Возможность параллельного выполнения нескольких задач с разным профилем прохождения на одних и тех же вычислительных ядрах

Архитектура ППВС позволяет в многозадачном режиме распараллеливать задачи на одни и те же вычислительные ядра. Аппаратура обеспечивает независимость выполнения таких задач друг от друга. Такая возможность востребована, когда выполняются задачи с разным профилем прохождения, например, одной задаче требуется больше ассоциативной памяти, а другой – больше функциональных устройств.

Кроме всего прочего, аппаратура позволяет начинать вычисления, не дожидаясь полного прихода всех начальных данных задачи.

IV. Заключение

В традиционных вычислительных системах кластерного типа проблема эффективного распараллеливания вычислений на большое число ядер решается в основном с помощью оптимизирующих компиляторов. Для распараллеливания задачи даже на сотни ядер требуются большие усилия программистов, усложняются программы. Параллельная потоковая вычислительная система, которая реализует потоковую модель вычислений с ДФК, решает многие проблемы традиционных высокопроизводительных вычислительных систем.

Использование ассоциативной памяти в ППВС во всей полноте реализует базовый принцип управления потоком данных, заключающийся в активации вычислений по готовности данных.

Средства распределения вычислений по пространству и во времени с использованием процедуры хэширования ключей токенов позволяют добиться равномерной загрузки вычислениями ИУ и исключить возможность переполнения ресурсов системы.

Простота создания параллельных программ, аппаратное управление памятью, аппаратное экстрагирование неявного параллелизма из программы, высокая степень масштабируемости задач и другие особенности потоковой модели вычислений и архитектуры ее реализующей делают эту модель вычислений перспективной для создания на ее основе будущих суперкомпьютерных систем, обладающих предельной производительностью [16].

По сравнению с предыдущими публикациями авторов по данной тематике в статье приведен наиболее полный перечень свойств потоковой модели вычислений с динамически формируемым контекстом и краткое описание особенностей аппаратной реализации этой модели, которые раскрывают преимущества этой модели вычислений в сопоставлении с традиционными подходами для создания будущих высокопроизводительных вычислительных систем.

Эксперименты [17]-[18], проведенные с использованием эмулятора ППВС, функционирующем на кластерном суперкомпьютере «Ломоносов» [19], и поведенческой модели на широком круге задач, свидетельствуют о высокой реальной производительности вычислительной системы и высокой степени масштабируемости выполняемых программ.

В частности, на задаче молекулярной динамики [17] было получено оптимальное количество обраба-

тываемых частиц на одно ядро, при котором масштабирование не падает, и составляет для ППВС 10^3 частиц, что на один-два порядка превышает тот же показатель для традиционных систем.

ЛИТЕРАТУРА

[1] URL: https://servernews.ru/967626 (дата обращения: 19.04.2018)

[2] Rampp M.. High-performance computing – technological trends and programming challenges // Max Planck Computing and Data Facility. URL: http://www.mpcdf.mpg.de/about-mpcdf/groups/ISSS12_Rampp_HPCtrends.pdf (дата обращения: 19.04.2018)

[3] Дикарев Н.И., Шабанов Б.М., Шмелёв А.С. Использование мелко гранулярного параллелизма в процессоре с архитектурой управления потоком данных // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2016. №2. С. 144-150.

[4] Hyperion (IDC) Paints a Bullish Picture of HPC Future. URL: https://www.hpcwire.com/2017/04/20/hyperion-idc-paints-bullish-picture-hpc-future/ (дата обращения: 19.04.2018)

[5] URL: http://www.r-ccs.riken.jp/en/overview/researchdiv/processor-research-team/ (дата обращения: 19.04.2018)

[6] Ким А.К., Перекатов В.И., Фельдман В.М. На пути к российской экзасистеме: планы разработчиков аппаратно-программной платформы «Эльбрус» по созданию суперкомпьютера экзафлопсной производительности // Вопросы радиоэлектроники. 2018. № 2. С. 6–13.

[7] URL: http://www.cnews.ru/news/top/2017-03-28_bajkal_elektroniks_nashel_zakazchikov_na_100 (дата обращения: 19.04.2018)

[8] 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.

[9] 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.

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

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

[12] 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.

[13] 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.

[14] Змеев Д.Н., Климов А.В., Левченко Н.Н., Окунев А.С., Стемпковский А.Л. Потоковая модель вычислений как парадигма программирования будущего // Информатика и её применения. 2015. Т. 9. Вып. 4. С. 29-36.

[15] 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: 19.04.2018).

[16] 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.

[17] Змеев Д.Н., Климов А.В., Левченко Н.Н., Окунев А.С., Стемпковский А.Л. Программирование задачи «молекулярная динамика» в потоковой модели вычислений // Журнал «Информационные технологии». 2017. №12. Т. 23. С. 859–867.

[18] Климов А.В., Поликарпов М.И. Опыт программирования одной задачи квантовой теории поля для потокового суперпроцессора ППВС «БУРАН» // XIV Международная Конференция «Супервычисления и математическое моделирование». Тезисы. г. Саров, 1-5 октября 2012 года, РФЯЦ-ВНИИЭФ. С. 105-106.

[19] Воеводин Вл.В., Жуматий С.А., Соболев С.И., Антонов А.С., Брызгалов П.А., Никитенко Д.А., Стефанов К.С., Воеводин Вад.В. Практика суперкомпьютера "Ломоносов" // Открытые системы. Москва: Издательский дом "Открытые системы", 2012. № 7. С. 36-39.

Advantages of Dataflow Computing Model

N.N. Levchenko, A.S. Okunev, A.L. Stempkovsky

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

***Abstract* — The main development trends in the field of high-performance computing demonstrate the growth of the number of computational cores to hundreds of thousands and even millions. At the same time, a high level of general concurrency of program execution should be maintained. To solve these issues, research and development of new architectures with the use of non-traditional computing models are conducted. This article describes the architecture of the parallel dataflow computing system (PDCS) that implements the original dataflow computing model with a dynamically formed context which is embodied in the parallel programming language DFL. The basic principles of the computing model are described: the operation of program nodes, their activation by data readiness; ways of token formation; the**

operation of the content addressable memory of keys; the distribution of computations using hashing, etc. The article also describes the main advantages of the computing model and the architecture of the PDCS, which allow creating well-scalable parallel programs for execution on hundreds of thousands of computational cores. First of all, it is the simplicity of creating parallel programs, thanks to which it is possible (for this architecture) to create and debug a program on one computational core, and then the definitive program will be automatically parallelized to any number of cores. The work with memory at the level of interaction between software nodes is "hidden" from the programmer; there is no problem of cache coherence. The PDCS architecture allows at runtime the identification of the task parallelism, which is initially present in the program and is implicit for the programmer. The use of the "scattering" paradigm provides a more efficient data migration strategy. By using hash functions, the programmer can adjust the degree of parallelism of the task execution and the degree of its locality, thereby reducing the loading of the communication network of the computing system. The features of the dataflow computing model presented in the article and their implementation in the architecture of the computing system make it possible to solve the problem of efficient parallelization of tasks on a large number of computational cores, which indicates the prospects of using this computing model for creating supercomputer systems with peak performance.

***Keywords* — dataflow computing model, architecture, advanced computing, computation scaling.**

REFERENCES

[1] Uskoritel' NVIDIA Quadro GV100 operiruet 32 Gbajt pamjati HBM2 - The NVIDIA Quadro GV100 accelerator operates on 32 GB of HBM2 memory. Available at: https://servernews.ru/967626 (accessed: 19.04.2018).

[2] Rampp M.. High-performance computing – technological trends and programming challenges. Max Planck Computing and Data Facility. Available at: http://www.mpcdf.mpg.de/about-mpcdf/groups/ISSS12_Rampp_HPCtrends.pdf (accessed: 19.04.2018).

[3] Dikarev N.I., Shabanov B.M., Shmeljov A.S. The use of fine-grained parallelism in dataflow processor. Problemy razrabotki perspektivnyh mikro- i nanojelektronnyh sistem (MJeS), 2016, no. 2, pp. 144-150 (in Russian).

[4] Hyperion (IDC) Paints a Bullish Picture of HPC Future. Available at: https://www.hpcwire.com/2017/04/20/hyperion-idc-paints-bullish-picture-hpc-future/ (accessed: 19.04.2018).

[5] Developing Parallel-computing Models and Acceleration Technologies for Large-scale High-performance Computing. Available at: http://www.r-ccs.riken.jp/en/overview/researchdiv/processor-research-team/ (accessed: 19.04.2018).

[6] Kim A.K., Perekatov V.I., Fel'dman V.M. On the way to russian exasistemes: plans of the Elbrus hardwaresoftware platform developers on creation of an exaflops performance supercomputer. Voprosy radiojelektroniki, 2018, no. 2, pp. 6–13 (in Russian).

[7] Voejkov D. Nachalos' krupnoserijnoe proizvodstvo rossijskih processorov «Bajkal» - Large-scale production of Russian processors "Baikal" began. Available at: http://www.cnews.ru/news/top/2017-03-28_bajkal_elektroniks_nashel_zakazchikov_na_100 (accessed: 19.04.2018).

[8] 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).

[9] 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).

[10] Lee B., Hurson A.R. Issues in Dataflow Computing. Advances in computers, 1993, vol. 37, pp. 285-333.

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

[12] 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.

[13] 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).

[14] Zmeev D.N., Klimov A.V., Levchenko N.N., Okunev A.S., Stempkovskij A.L. Dataflow computing model as a paradigm of future mainstream of software development. Informatika i ejo primenenija, 2015, vol. 9, no. 4, pp. 29-36 (in Russian).

[15] Klimov A.V., Levchenko N.N., Okunev A.S., Stempkovskij A.L. Supercomputers, memory hierarchy and dataflow computing model. Programmnye sistemy: teorija i prilozhenija: jelektron. 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: 18.04.2018).

[16] 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).

[17] Zmeev D.N., Klimov A.V., Levchenko N.N., Okunev A.S., Stempkovskij A.L. Programming of molecular dynamics task in dataflow computing model. Zhurnal «Informacionnye tehnologii», 2017, no. 12, vol. 23, pp. 859–867 (in Russian).

[18] Klimov A.V., Polikarpov M.I. Experience of programming one problem of quantum field theory for dataflow superprocessor PDCS "Buran". XIV Mezhdunarodnaja Konferencija «Supervychislenija i matematicheskoe modelirovanie» - Proc. XIV Int. Conf. “Supercomputing and mathematical modeling”. Sarov, 2012, pp. 105-106 (in Russian).

[19] Voevodin Vl.V., Zhumatij S.A., Sobolev S.I., Antonov A.S., Bryzgalov P.A., Nikitenko D.A., Stefanov K.S., Voevodin Vad.V. Practice of "Lomonosov" Supercomputer. Otkrytye sistemy. Moskva: Izdatel'skij dom "Otkrytye sistemy", 2012, no. 7, pp. 36-39 (in Russian).