МИКРОЭЛЕКТРОНИКА, 2019, том 48, № 5, с. 343–350

ИЗБРАННЫЕ МАТЕРИАЛЫ КОНФЕРЕНЦИИ “ПРОБЛЕМЫ РАЗРАБОТКИ ПЕРСПЕКТИВНЫХ МИКРО- И НАНОЭЛЕКТРОННЫХ СИСТЕМ” ПРОЕКТИРОВАНИЕ ИНТЕГРАЛЬНЫХ СХЕМ

УДК 004.27

ОСОБЕННОСТИ АРХИТЕКТУРЫ, РЕАЛИЗУЮЩЕЙ ПОТОКОВУЮ МОДЕЛЬ ВЫЧИСЛЕНИЙ, И ЕЕ ПРИМЕНЕНИЕ ПРИ СОЗДАНИИ МИКРОЭЛЕКТРОННЫХ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

© 2019 г. Д. Н. Змеев[1], А. В. Климов[1], Н. Н. Левченко[1], А. С. Окунев[1, *], А. Л. Стемпковский[1]

[1]Институт проблем проектирования в микроэлектронике Российской АН, ул. Советская, 3, Зеленоград, Москва, 124365 Россия

*****E-mail: oku@ippm.ru*

Поступила в редакцию 03.04.2019 г.
После доработки 03.04.2019 г.
Принята к публикации 03.04.2019 г.

В настоящее время высокопроизводительные вычисления используются повсеместно: в государственном секторе, оборонной сфере, крупных концернах и предприятиях, коммерческих структурах и др. При этом с каждым годом, все больше средств вкладывается в суперкомпьютеры и программное обеспечение для того, чтобы быстро отвечать на любые “вызовы” в своих областях. Современные высокопроизводительные вычислительные системы в общей массе базируются на фоннеймановской модели вычислений. Это в свою очередь накладывает определенные ограничения, в том числе и на распараллеливание вычислений, и как следствие – на создание эффективных параллельных программ. В статье предлагается один из вариантов решения этих проблем – переход к оригинальной потоковой модели вычислений с динамически формируемым контекстом и ее реализации в параллельной потоковой вычислительной системе. Описываются особенности архитектуры параллельной потоковой вычислительной системы и предоставляемые ими преимущества. Подобные архитектуры будут востребованы при создании новых микроэлектронных высокопроизводительных вычислительных систем.

DOI: 10.1134/S0544126919050119

ВВЕДЕНИЕ

В области высокопроизводительных вычислений имеется потребность в более эффективном использовании аппаратных ресурсов современных суперкомпьютеров при решении актуальных задач. В настоящее время в традиционных вычислительных системах с увеличением числа процессоров падает реальная производительность отдельного процессора. К тому же архитектура современных суперкомпьютеров не обеспечивает хорошее масштабирование актуальных задач со сложноорганизованными данными и нерегулярной структурной организацией. Это связано с использованием традиционной (фон-неймановской) модели вычислений и архитектур, базирующихся на этой модели. Наряду с этим постоянно увеличивается число алгоритмов, которые не могут эффективно распараллеливаться на суперкомпьютерных системах кластерного типа. Надо также отметить, что потенциал многопроцессорных и многоядерных вычислительных систем может быть реализован только с использованием параллельного программного обеспечения. В свою очередь, параллельное программирование и трудности, связанные с ним, являются для всех производителей компьютеров большой проблемой, которую они вынуждены решать совместно с создателями программного обеспечения.

Все эти проблемы, связанные с эффективной загрузкой аппаратных ресурсов суперкомпьютера, ростом энергопотребления, сложностью параллельного программирования и ряд других, предлагается решать посредством перехода к потоковой модели вычислений с динамически формируемым контекстом.

В восьмидесятые годы прошлого столетия за рубежом велись работы над архитектурами вычислительных систем, реализующих потоковую модель вычислений – dataflow [1, 2]. Модель вычислений с управлением потоком данных по сравнению с фон-

Сокращения: АП – ассоциативная память; ИУ – исполнительные устройства; ППВС – параллельная потоковая вычислительная система; БХФЭ – блоки хэш-функции подмножеств (этапов); ПС – процессор сопоставления; АПК – ассоциативная память ключей; ВМ – вычислительный модуль.

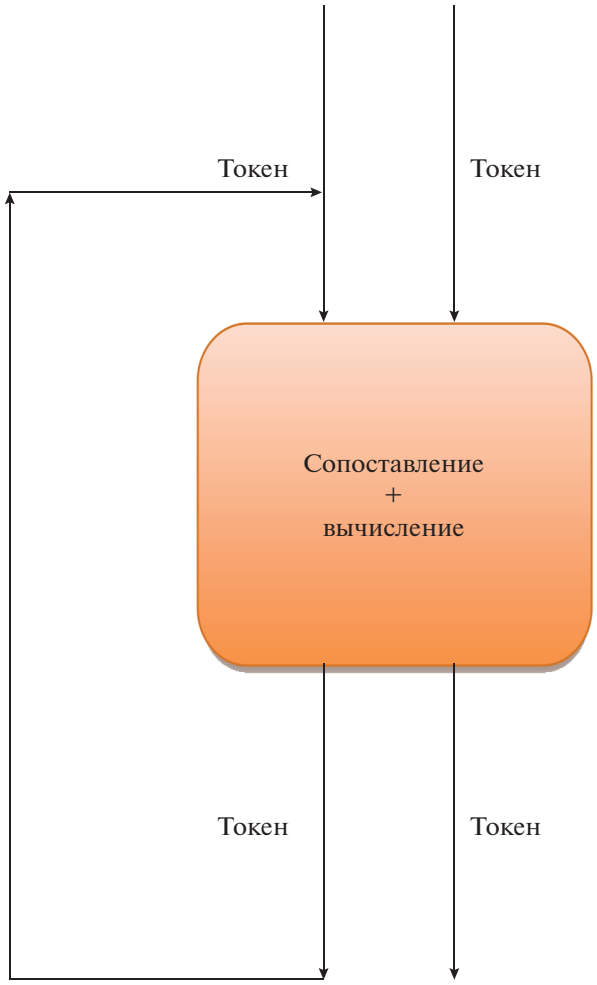

Рис. 1. Потоковая модель вычислений.

неймановской моделью задумывалась как более адекватная аппаратуре, в частности, разработчики утверждали, что она значительно лучше использует присущий аппаратуре “естественный” параллелизм [3]. Однако, в дальнейшем основные усилия разработчиков были направлены на приведение модели вычислений в соответствие сложившейся на тот момент методологии программирования: разработке адекватных ей языков программирования высокого уровня с привычной нотацией, подгонке аппаратной реализации как под эти новые, так и старые языки. В результате конкурентные преимущества были утрачены, что в конце концов и привело к фактическому сворачиванию всех работ по данному направлению за рубежом [3]. В последние годы и зарубежные разработчики суперкомпьютерных вычислительных систем, осознав преимущества потоковой модели вычислений, возобновили исследования в области нетрадиционных подходов для их применения в экстрамасштабных вычислениях (Extreme-Scale Computing) [4, 5], проводятся конференции [6], публикуются статьи.

В России работы над потоковой моделью вычислений были начаты в ИВВС РАН и ИПИ РАН, а с 2007 года – продолжены в ИППМ РАН. Потоковая модель вычислений с динамически формируемым контекстом и реализующая ее архитектура обладают большим потенциалом в области создания новых микроэлектронных высокопроизводительных вычислительных систем петафлопсного и сверхпетафлопсного уровня производительности. В статье рассматривается этот потенциал с точки зрения особенностей архитектуры вычислительной системы.

1. МОДЕЛЬ ВЫЧИСЛЕНИЙ

В основе модели вычислений с управлением потоком данных (или потоковой модели вычислений) лежит принцип сопоставления данных. Схематично работа потоковой модели вычислений представлена на рис. 1. На вход абстрактного устройства сопоставления и вычисления поступают данные в виде токенов. Токен представляет собой структуру, содержащую помимо самого данного (операнда), еще и ключ, в состав которого входят: контекст (индекс), который однозначно определяет положение данного в виртуальном адресном пространстве задачи, адрес программы узла (или код операции), обрабатывающего данные, а также ряд служебных полей. В абстрактном устройстве происходит сопоставление токенов между собой по определенным правилам. Как только находится комбинация токенов, удовлетворяющая всем условиям, происходит обработка операндов, присутствующих в токенах согласно программе узла (или коду операции). Результатом такой обработки являются новые токены, которые либо поступают на вход этого абстрактного устройства, либо выдаются наружу в качестве результирующих данных.

В ИППМ РАН ведутся исследования потоковой модели вычислений с динамически формируемым контекстом [7]. Ключевым ее отличием от традиционных потоковых моделей вычислений является наличие динамического формирования контекста.

В качестве абстрактного устройства сопоставления и вычисления в потоковой модели вычислений с динамически формируемым контекстом (рис. 2) выступают АП и набор исполнительных устройств. Ассоциативная память отвечает за сопоставление данных по определенным законам и формирование, так называемого, пакета, представляющего собой структуру, содержащую операнды токенов, ключ, контекст и ряд служебных полей. Процесс сравнения (сопоставления) одного данного одновременно с множеством других

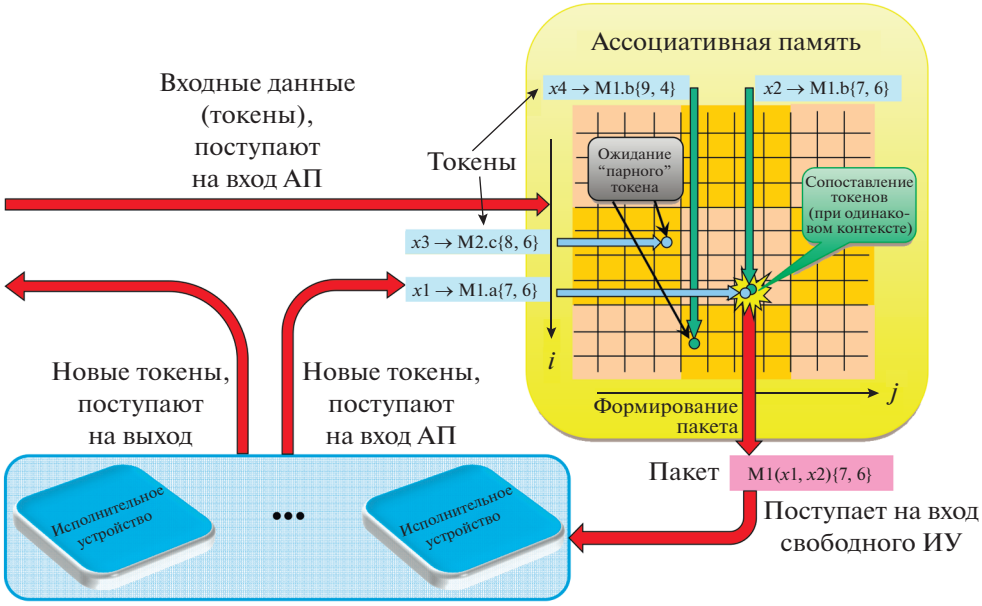

Рис. 2. Потоковая модель вычислений с динамически формируемым контекстом.

данных является имманентным свойством ассоциативной памяти, что и послужило поводом для выбора АП в качестве основного сопоставляющего устройства.

Далее пакет обрабатывается в исполнительном устройстве. Результатом обработки этого пакета являются новые токены, которые поступают либо в ассоциативную память, либо подаются на выход.

Еще одним отличием (в том числе и от классической фон-неймановской модели вычислений) является использование парадигмы “раздачи” (рис. 3) при программировании и выполнении задачи в потоковой модели вычислений. В парадигме “сбора” источник нового данного сохраняет свой результат в памяти по некоторому адресу, по которому впоследствии его запрашивает потребитель (по мере надобности). В парадигме “раздачи” источник нового данного (узел) “знает” (вычисляет) адреса всех его потребителей, на которые и происходит рассылка результата работы узла.

Использование парадигмы “раздачи” позволяет получить следующие преимущества: во-первых, наличие информации о будущем использовании данных, что позволяет управлять своевременной подкачкой данных; во-вторых, автоматическое удаление из памяти данных после своего использования; в-третьих, сокращение числа обращений к данному.

Потоковая модель вычислений с динамически формируемым контекстом реализуется в архитектуре параллельной потоковой вычислительной системы (ППВС) “Буран”.

2. АРХИТЕКТУРА ППВС

Архитектура параллельной потоковой вычислительной системы [8] представляет собой набор объединенных глобальным коммутатором вычислительных модулей. К этому коммутатору подключен процессор ввода/вывода [9], через который осуществляется связь с “внешним миром” (рис. 4).

Каждый вычислительный модуль конструктивно состоит из следующих основных узлов и блоков (рис. 5):

– внутримодульного коммутатора токенов;
– блока хэш-функции подмножеств (этапов);
– процессора сопоставления;
– формирователя пакетов;
– внутримодульного коммутатора пакетов;
– исполнительного устройства;
– блока локализующей хэш-функции.

Внутримодульный коммутатор токенов предназначен для перенаправления токенов между разными вычислительными ядрами и глобальным коммутатором токенов. Блоки хэш-функции этапов и локализации предназначены для распределения вычислений по пространству и во времени.

Работа вычислительного модуля ППВС заключается в следующем. Токен, поступая из глобального коммутатора токенов, через внутримодульный коммутатор токенов поступает согласно содержимому поля “№ ПС” (номер процессора сопоставления) в один из блоков хэш-функции этапов, который напрямую связан с процессором сопоставления, имеющим уникальный номер. Процессор сопоставления, являющийся основ-

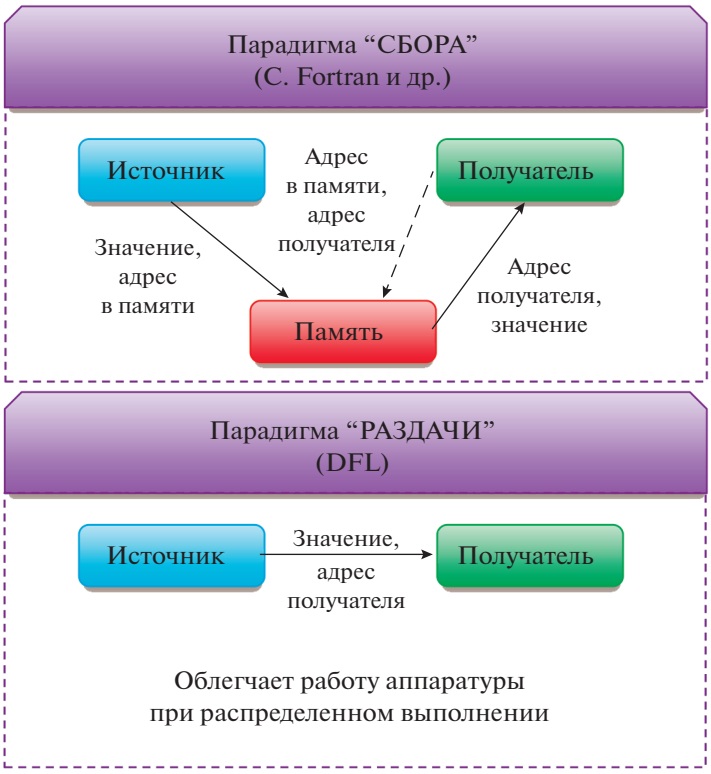

Рис. 3. Сравнение парадигм “сбора” и “раздачи”.

ным блоком ППВС, состоит из следующих основных узлов: ассоциативной памяти, буферов токенов и пакетов, устройства управления, узла сбора статистики и формирователя пакетов.

В блоке хэш-функции этапов пришедший токен получает номер этапа, к которому он принадлежит, и поступает на вход процессора сопоставления. Этапы в ППВС могут быть “активными” и “пассивными”. Токены “пассивных” этапов получают статус “отложенных”. Чтобы эти токены обрабатывались, этап должен получить статус “активный”.

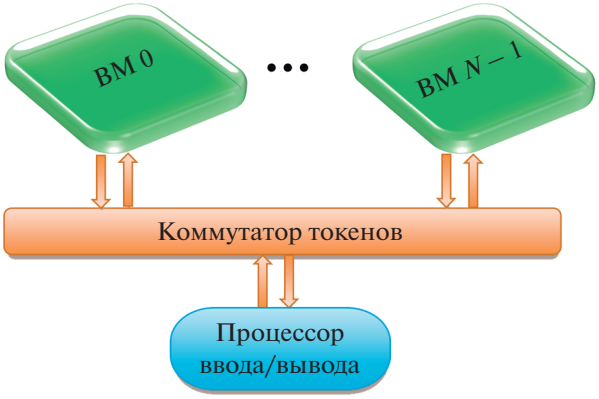

Рис. 4. Архитектура ППВС “Буран”.

В процессоре сопоставления выполняется проверка на принадлежность токена “активному” этапу, иначе он пересылается в память отложенных токенов. “Активный” токен поступает на вход ассоциативной памяти ключей, реализованной аппаратно. С целью экономии ассоциативной памяти, она конструктивно разбита на несколько блоков – ассоциативная память ключей, память дескрипторов и память токенов, причем память дескрипторов и память токенов выполнены с использованием прямоадресуемой памяти. В АПК происходит сопоставление входного токена с ранее пришедшими токенами по ключу. Если будет найден токен, “совпадающий” с входным, то оба токена передаются в формирователь пакетов. Пакет через внутримодульный коммутатор пакетов поступает на вход любого свободного исполнительного устройства.

Особо стоит отметить, что обработка пакета не привязана к конкретному ИУ, поскольку каждый пакет в своем составе содержит номер программы узла, которая обрабатывает данные, содержащиеся в пакете, а каждое исполнительное устройство содержит всю программу решаемой задачи.

Результатом обработки пакета являются новые токены, которые поступая в блок локализующей функции, получают номер ПС, в который они должны будут поступить. Этот процесс повторяется до окончания вычислений.

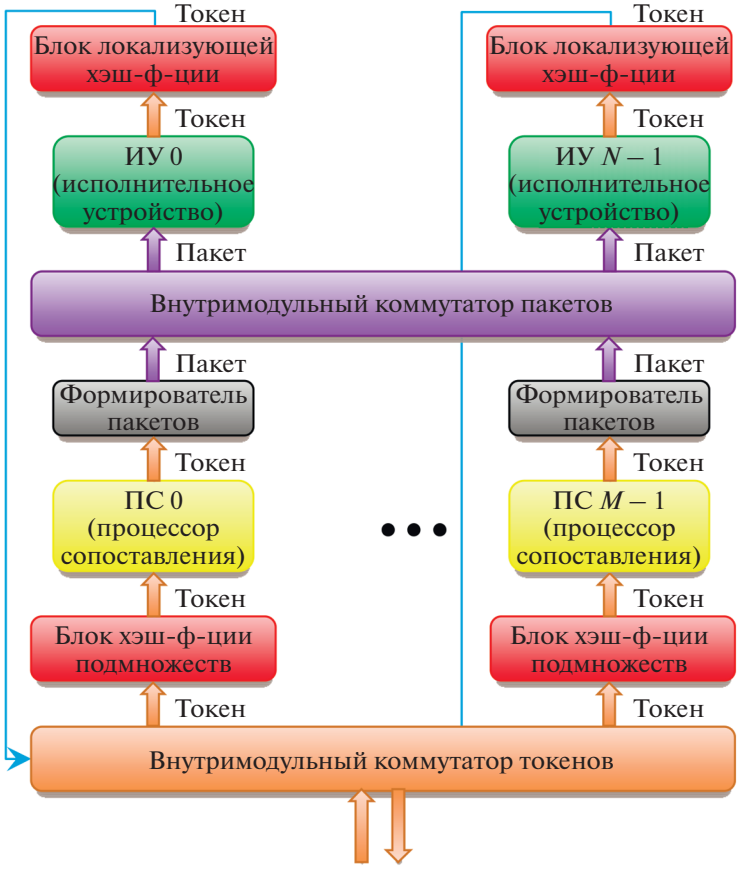

Рис. 5. Архитектура вычислительного модуля ППВС “Буран”.

3. ОСОБЕННОСТИ АРХИТЕКТУРЫ ППВС

Архитектура ППВС отличается от архитектур традиционных высокопроизводительных вычислительных систем. Эти отличия и особенности, которые позволяют решить проблемы построения и, главное, эффективного использования суперкомпьютерных систем, кратко описываются в данном разделе статьи.

Аппаратное управление вычислительными ресурсами

В традиционных многопроцессорных вычислительных системах, для эффективного использования всех имеющихся аппаратных ресурсов, программистам приходится много сил уделять тончайшей настройке программы и самого алгоритма под конкретную систему (число ядер в процессоре и число процессоров в модуле, размер кэшей, тип коммутационной сети и др.). В параллельной потоковой вычислительной системе многие из этих вещей скрыты от программиста. В частности, управление распределением памяти ППВС происходит без участия программиста (на аппаратном уровне). Программа создает токены, которые распределяются вычислительной системой для сопоставления или хранения (в случае ожидания “парного” токена) на свободном месте в ассоциативной памяти автоматически. При этом отметим, что управление распределением вычислений не связано с текстом программы. Это позволяет один раз запрограммировать задачу, а затем менять (или настраивать) только распределение (функцию локализации) вычислений, не внося изменений в отлаженную программу.

В ППВС распределение вычислений по пространству (по вычислительным ядрам) происходит при помощи использования хэш-функций, часть из которых реализованы на аппаратном уровне. Однако программист при желании под свою задачу может создать собственную высокоэффективную функцию распределения и задать ее выполнение программно. Входными данными для таких хэш-функций является исключительно ключ токена (содержимое его отдельных полей). На основе этих данных хэш-функция вырабатывает значение – номер процессора сопоставления (вычислительного ядра), в который должен будет поступить токен. Причем токены с одинаковыми полями ключа получат одинаковый номер про-

цессора сопоставления, что обеспечит их “встречу” в одном процессоре сопоставления.

Используя механизм хэширования, программист может добиться не только лучшего распределения данных по вычислительным ядрам, но и уменьшить число передач (посылок токенов) между ними, сосредоточив вычисления внутри вычислительного ядра. Это позволяет снизить нагрузку на коммуникационную сеть вычислительной системы и тем самым повысить эффективность выполнения задачи.

Аппаратное экстрагирование неявного параллелизма

Особенности потоковой модели вычислений с динамически формируемым контекстом и самой архитектуры ППВС, в которой она реализуется, позволяют непосредственно в процессе вычисления выявлять так называемый неявный параллелизм на нескольких уровнях выполнения программы:

– между итерациями;

– между разными активациями программных узлов;

– между разными программными узлами;

– между разными подзадачами (стадиями).

Неявный параллелизм изначально заложен в самой программе, но для программистов сложно заранее его выявить и явно использовать. Аппаратура ППВС позволяет это делать благодаря свойствам ассоциативной памяти ключей, подготавливающей данные, которые можно затем обрабатывать на исполнительных устройствах системы.

Аппаратная поддержка мелкозернистого параллелизма

В параллельной потоковой вычислительной системе изначально была реализована аппаратная поддержка организации активации программного узла на уровне отдельных данных-токенов (программа состоит из десятков-сотен операций) над одним набором данных. Позже была добавлена возможность укрупнения программных узлов (в том числе с помощью введения многовходовых конструкций). Однако, чем крупнее программный узел, тем меньше уровень параллелизма всей программы. Поэтому, программист, фактически, через определение размера и количества программных узлов может задавать требуемый уровень параллелизма всей программы, который зависит как от входных данных, так и от имеющихся аппаратных ресурсов (например, числа вычислительных ядер).

Низкий семантический разрыв

Существует семантический разрыв между любой архитектурой вычислительной системы и средой программирования для нее. В современных системах этот разрыв – большой, что порождает следующие проблемы: увеличивается стоимость разработки программного обеспечения, падает его надежность, усложняется операционная система и разрабатываемые компиляторы.

Семантический разрыв между параллельным языком высокого уровня ППВС и архитектурой ППВС существенно ниже. Это позволяет обеспечить рост надежности программного обеспечения, повысить продуктивность программирования и эффективность использования аппаратуры, сократить время решения задачи, а также снизить стоимость создания программного обеспечения.

Работа с неполным набором данных

Еще одной особенностью архитектуры ППВС является возможность начинать вычисления с приходом первых токенов, не дожидаясь формирования всего массива входных данных. Это открывает новые возможности по ускорению вычислений, определяя порядок подачи входных данных, что позволяет эффективно использовать данную архитектуру в системах, работающих в реальном масштабе времени. В момент прихода “последнего” данного (операнда) часть работы может быть уже выполнена, при условии отсутствия зависимостей по данным в программе от этого операнда.

Совмещение вычислений и обменов данными

Принцип “decouple access execute” компьютерной архитектуры был предложен еще в 1982 году Дж. Смитом [10]. Суть этого принципа, актуального и для новых архитектур [11], заключается в разделении стадии вычислений и стадии подготовки данных для них. Этот принцип является неотъемлемой частью самой потоковой модели вычислений и архитектуры ППВС, поскольку за подготовку данных отвечает процессор сопоставления, определяющий токены, готовые к обработке, а за выполнение вычислений над данными отвечает исполнительное устройство.

“Простые” исполнительные устройства

Одним из серьезных вызовов, стоящих перед создателями современных вычислительных систем является проблема уменьшения энергопотребления. Одним из способов его уменьшения с архитектурной точки зрения является использование простых (а не мультитредовых) исполнительных устройств. Это позволяет не тратить энергию на исполнение спекулятивных вычислений, а так-

же на переключение тредов. А благодаря особенностям архитектуры вычислительного модуля ППВС число исполнительных устройств, работающих с одним процессором сопоставления, может легко изменяться в зависимости от класса выполняемых программ.

Следует отметить, что выполнение программы узла в исполнительном устройстве не прерывается на дополнительную подкачку "внешних данных". Программный узел оперирует входными данными присутствующими в полях ключа, а также константами. Никаких дополнительных данных не требуется, что позволяет выполнять программу узла без прерывания.

Уменьшение числа сопоставлений в АП

Ассоциативная память считается более энергозатратной по сравнению с прямоадресуемой памятью. Это было одной из причин того, что в конце 80-х годов прошлого века исследования по направлению dataflow были свернуты. В процессе работы над архитектурой ППВС были предложены и реализованы в программных моделях ППВС различного уровня абстракции аппаратные и программные методы уменьшения числа сопоставлений в ассоциативной памяти ключей процессора сопоставления, что приводит к уменьшению энергопотребления [12]. В число таких методов входит – разделение ассоциативной памяти на память ключей (ассоциативную) и память токенов (прямоадресуемую), поэтапная обработка сопоставления ключей и ряд других.

Параллельное выполнение нескольких задач

Архитектура параллельной потоковой вычислительной системы позволяет в многозадачном режиме выполнять разные задачи на одном и том же вычислительном ядре, это также справедливо и в отношении распараллеливания разных задач на одну и ту же группу вычислительных ядер. Аппаратура системы обеспечивает отсутствие взаимодействия между такими задачами во время выполнения. Это особенно актуально в том случае, когда у задач разный профиль прохождения, то есть, к примеру, одной задаче требуется больше вычислительных ресурсов, другой – больше памяти, а третьей – интенсивный обмен по коммуникационной сети.

Высокая степень масштабируемости задач

Одним из основных особенностей и преимуществ архитектуры ППВС является высокая степень масштабируемости задач, а также то, что при увеличении числа вычислительных ядер падение реальной производительности происходит существенно медленнее, чем при решении тех же задач на традиционных системах. Особенно это заметно на задачах со сложноорганизованными или многомасштабными данными. Этому способствует следующее:

– активация программного узла по готовности данных;

– локальность программного узла (активируемому узлу для своего полного выполнения не требуется никаких дополнительных данных;

– независимость выполнения программного узла (от других программных узлов);

– один и тот же протокол взаимодействия (между вычислительными модулями и внутри них действует единый протокол, осуществляющий доставку токенов к ядрам).

Стоит также отметить, что программа, написанная для одного ядра, без изменений работает на любом количестве ядер. Все эти особенности архитектуры ППВС, реализующей потоковую модель вычислений, в будущем позволят найти широкое применение этой архитектуре при создании микроэлектронных высокопроизводительных вычислительных систем как универсального, так и специального назначения.

ЗАКЛЮЧЕНИЕ

Проектирование современных суперкомпьютеров должно завершаться созданием эффективных, с точки зрения использования вычислительных ресурсов, многопроцессорных систем. Кроме того, должна быть возможность распараллелить на сотни тысяч и более вычислительных ядер актуальные задачи. По мнению авторов, для решения этих задач необходимо переходить к нетрадиционным моделям вычислений и архитектурам, основанным на этих моделях.

Создание вычислительной системы, реализующей потоковую модель вычислений с динамически формируемым контекстом, позволяет решить многие проблемы традиционных кластерных суперкомпьютерных систем. Особенности архитектуры ППВС описанные в статье, более ясно представляют различия классической и нетрадиционной моделей вычислений.

Эксперименты, которые проводились на широком круге задач при использовании поведенческой блочно-регистровой модели и эмулятора ППВС, функционирующего на кластерном суперкомпьютере "Ломоносов" [13], позволяют говорить о высокой реальной производительности параллельной потоковой вычислительной системы, а также о высокой степени масштабируемости выполняемых программ.

Простота создания параллельных программ, аппаратное управление памятью, аппаратное экстрагирование неявного параллелизма из программы, эффективная работа программы с неполным набо-

ром данных, 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.

СПИСОК ЛИТЕРАТУРЫ

1. *Lee B., Hurson A.R.* Issues in Dataflow Computing // Advances in computers. 1993. V. 37. P. 285–333.
2. *Lee B., Hurson A.R.* Dataflow Architectures and Multithreading // Computer. Aug 1994. V. 27. № 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. V. 1. № 1. P. 3–30.
4. *Sebastian Ertel, Justus Adam, Jeronimo Castrillon.* Supporting Fine-grained Dataflow Parallelism in Big Data Systems // In Proceedings of the 9th International Workshop on Programming Models and Applications for Multicores and Manycores (PMAM'18), Chen Q., Huang Z., and Balaji P. (Eds.). ACM, New York, NY, USA. 2018. P. 41–50. https://doi.org/10.1145/3178442.3178447
5. *Tony Nowatzki, Vinay Gangadhar, Newsha Ardalani, Karthikeyan Sankaralingam.* Stream-Dataflow Acceleration // In Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA'17). ACM, New York, USA. 2017. P. 416–429. https://doi.org/10.1145/3079856.3080255
6. International Workshop on Data Flow Models for Extreme-Scale Computing (DFM 2017). URL: http://www.cs.ucy.ac.cy/dfmworkshop/ (дата обращения: 28.10.2018).
7. *Климов А.В., Левченко Н.Н., Окунев А.С.* Преимущества потоковой модели вычислений в условиях неоднородных сетей // Журн. "Информационные технологии и вычислительные системы". 2012. № 2. С. 36–45.
8. *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.
9. *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
10. *Smith James E.* Decoupled access/execute computer architectures. In Proceedings of the 9th annual symposium on Computer Architecture (ISCA'82). IEEE Computer Society Press, Los Alamitos, CA, USA. P. 112–119.
11. *George Charitopoulos, Charalampos Vatsolakis, Grigorios Chrysos, Dionisios N. Pnevmatikatos.* A decoupled access-execute architecture for reconfigurable accelerators. In Proceedings of the 15th ACM International Conference on Computing Frontiers (CF'18). ACM, New York, NY, USA. P. 244–247. https://doi.org/10.1145/3203217.3203267
12. *Levchenko N.N., Okunev A.S., Yakhontov D.E., Zmejev D.N.* Decreasing the Power Consumption of Content-Addressable Memory in the Dataflow Parallel Computing System // IEEE EAST-WEST DESIGN & TEST SYMPOSIUM 2012, Kharkov, Ukraine, September 14–17, 2012, Symposium Proceedings. P. 122–125.
13. *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. М.: Издательский дом "Открытые системы". 2012. № 7. С. 36–39.