УДК 004.27

# Использование модели вычислений с управлением потоком данных и реализующей ее архитектуры для систем эксафлопсного уровня производительности

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

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

***Аннотация* — В статье рассматривается использование модели вычислений с управлением потоком данных и реализующей ее архитектуры для организации вычислительного процесса в системах эксафлопсного уровня производительности. Кроме того, приводятся основные пути увеличения производительности компьютерных систем, описывается архитектура параллельной потоковой вычислительной системы и архитектура базового вычислительного модуля системы.**

***Ключевые слова* — процессор сопоставления, модель вычислений с управлением потоком данных, эксафлопсная производительность.**

## I. Введение

В настоящее время перед создателями суперкомпьютерных систем возник очередной вызов – создание эксафлопсной вычислительной системы. В настоящее время используются следующие методы повышения производительности вычислительных систем: увеличение тактовой частоты; увеличение ядер на кристалле (процессоре), поскольку производительность одного ядра на используемой элементной базе близка к пределу; применение коммерчески доступных компонентов для создания кластеров и другие методы. Экспертами признано, что эти методы не применимы для создания эксафлопсных систем.

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

Потенциал многоядерных (многопроцессорных) вычислительных систем, а именно только такими и могут быть суперкомпьютеры, во всей полноте может использовать только параллельное программное обеспечение путем распараллеливания вычислительных процессов. Необходимость параллельного программирования и трудности, которые с ним связаны, также являются большой проблемой для всех производителей компьютеров.

На рис. 1 приведены основные пути увеличения производительности компьютерных систем. И для создания эксафлопсных систем придется использовать практически все возможности. Одним из направлений является создание все более совершенных базовых вычислительных (процессорных) модулей. К базовым вычислительным модулям можно отнести как платформы на базе универсальных процессоров от компаний Intel и AMD, так и на базе специализированных вычислителей, например, 100-ядерный процессор фирмы Tilera [1], Intel Knights Corner (входящий в платформу MIC – Many Integrated Core) [2], и «графические» ускорители от компаний Nvidia и AMD. Основной проблемой в использовании таких базовых вычислительных модулей является эффективная загрузка всех имеющихся аппаратных ресурсов одной задачей. Основным способом решения данной проблемы в настоящее время является использование оптимизирующих трансляторов, а также весьма «изощренное» программирование, требующее от пользователя серьезных навыков. И хотя компании добились в этом деле серьезных успехов, далеко не всегда можно в статике эффективно распределить нагрузку на имеющиеся аппаратные средства.


**Рис. 1. Пути увеличения производительности компьютерных систем**

Проблему эффективной загрузки аппаратных ресурсов высокопроизводительных систем, как на кристалле, так и в целом в вычислительной системе, предлагается решать, используя модель вычислений с

управлением потоком данных и динамически формируемым контекстом.

## II. Потоковая модель вычислений

Потоковые модели вычислений начали разрабатываться в мире с 70-х годов прошлого века. Примерами таких моделей и реализующих их прототипов вычислительных систем являются MIT Tagged-Token Dataflow Architecture, проект Monsoon, StarT, WaveScalar и TRIPS.

Однако в данных системах не был решен ряд принципиальных вопросов, сдерживающих развитие данного направления, таких как, например, планирование вычислений во времени, локализация вычислений по пространству и проблема создания иерархии памяти.

### *A. Модель вычислений с управлением потоком данных с динамически формируемым контекстом*

Предлагаемая модель вычислений основана на принципе управления потоком данных, когда активация вычислительных квантов осуществляется по готовности данных – токенов (структура данных, содержащая, помимо самого данного, служебную информацию и ключ, который однозначно определяет положение операнда в виртуальном адресном пространстве задачи). Под вычислительным квантом понимается программа, которая, будучи активированной, может быть доведена до конца без привлечения дополнительной информации. Различные вычислительные кванты между собой взаимодействуют и сохраняют состояние только через отправку токенов, которые активируют новые вычислительные кванты. Отправителем определяется как передаваемое значение, так и адрес получателя. В этом состоит принципиальное отличие нашего подхода от традиционного, когда вычислительный процесс сам запрашивает нужные ему данные (из памяти или у других процессов).

### *B. Динамически формируемый контекст*

В ранее создаваемых системах, основывающихся на вычислительных моделях, управляемых потоком данных, контекст присутствовал в качестве идентификатора данного или цикла, и был недоступен для изменения и использования пользователем в динамике.

Динамически формируемый контекст (ДФК) в нашей модели вычислений позволяет осуществлять одновременное и параллельное выполнение программы узла над различными вычислительными квантами, имеющими разный ключ. Ключ представляет собой часть токена, по которому процессор сопоставления осуществляет работу по поиску «совпадающих» токенов. Весь ключ полностью доступен для программиста. Использование ДФК позволяет фактически в динамике «настраивать» токен по месту, то есть программа узла получает всю необходимую информацию из ключа о местоположении токена, и, соответственно, знает, куда и какое число результирующих токенов послать в зависимости от выполняемых условий.

При программировании требуется правильно закодировать контекст (то есть разбить его на поля) и преобразовывать его в динамике. Граф программы часто зависит от данных (от размерности или кодировки контекста) и, благодаря ДФК, сам граф программы строится в динамике. ДФК позволяет повысить эффективность программирования и дает возможность реализовать новые принципы организации параллельных вычислительных процессов.

## III. Описание архитектуры параллельной потоковой вычислительной системы

Потоковую модель вычислений с динамически формируемым контекстом реализует параллельная потоковая вычислительная система (ППВС) «Буран» [3].

ППВС представляет собой многоядерную масштабируемую вычислительную систему (рис. 2). Между ядрами в системе передаются единицы информации в виде токенов. Коммутация между ядрами осуществляется на основе значения номера ядра, вырабатываемого блоком хэширования на основе настраиваемой функции распределения.

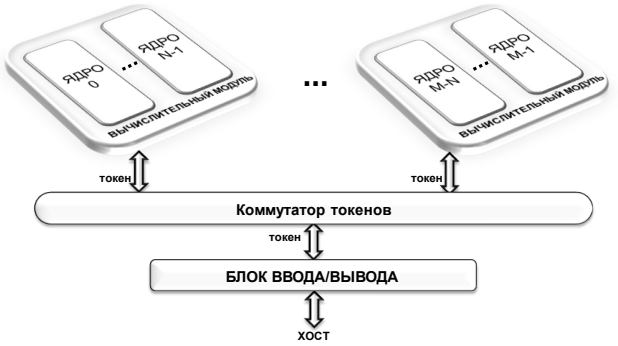


**Рис. 2. Структурная схема ППВС**

Вычислительные ядра (рис. 3) в пределах одного кристалла организуются в вычислительные модули. 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 принципа потока данных, когда готовые к выполнению данные активируют выполнение программы узла; во-вторых, активируемому узлу для своего полного выполнения не требуется никаких дополнительных данных; в-третьих, узлы выполняются полностью независимо друг от друга и, в-четвертых, благодаря правильному выбору хэш-функции (функции распределения вычислений по группам ядер), сокращающей число межъядерных передач токенов.

Благодаря такой модели вычислений, реализуемой в архитектуре ППВС, выполнение программы и ее создание не зависят от конкретной конфигурации вычислительной системы, то есть программа будет работать

и на одном ядре, и на любом другом числе ядер без повторной компиляции. Для улучшения прохождения программы на различных конфигурациях требуется только изменить параметры или саму хэш-функцию.

## IV. Архитектура вычислительного модуля

Вычислительный модуль (ВМ) системы конструктивно состоит из набора вычислительных ядер (фактически являясь многоядерным процессором), в состав которых входят:

- процессор сопоставления (ПС), в котором происходит сравнение токенов по определенным правилам и формирование пакетов (структуры данных, которая содержит операнды, необходимые для активации вычислительного кванта – программы узла);

- исполнительное устройство (ИУ), в котором в соответствии с программой узла происходит обработка пакета и генерация новых токенов;

- блок хэш-функции, в котором осуществляется определение номера ядра, куда передается сформированный токен из исполнительного устройства;

- внутренний коммутатор токенов (ВКМ), который передает токен либо во внешнюю сеть, либо в свой ПС;

- внутренний коммутатор пакетов, который передает пакет на любое свободное ИУ вычислительного модуля.

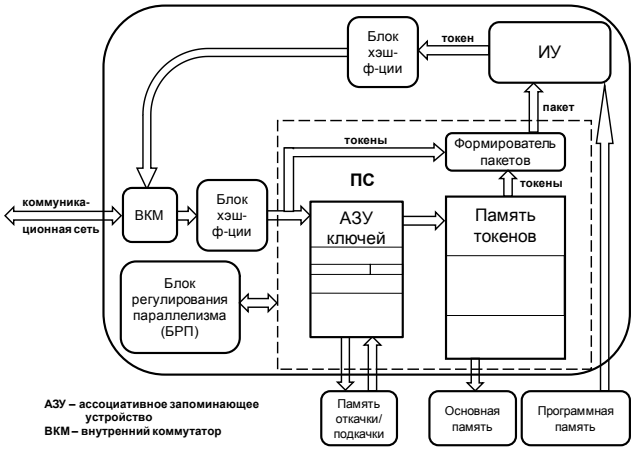


**Рис. 3. Функциональная схема вычислительного ядра**

### A. *Процессор сопоставления*

Основным узлом вычислительного ядра ППВС «Буран», реализующим базовые принципы модели вычислений с управлением потоком данных, является процессор сопоставления со своей системой команд. Процессор сопоставления содержит сопоставляющее устройство, которое реализуется в виде аппаратной ассоциативной памяти, разбитой на модули. Основными функциями ПС являются:

- организация работы вычислительных процессов и синхронизация их по данным;

- прием, сопоставление приходящих токенов и создание пакетов из совпавших токенов;

- распределение вычислительных ресурсов с помощью управляющих операций;

- поддержание баланса загрузки вычислительных ресурсов с использованием механизмов откачки/подкачки, пересчета кратности и др.;

- аппаратная поддержка многовходовых и векторных узлов, что увеличивает производительность системы, благодаря увеличению числа выполняемых операций за такт и снижению доли накладных расходов при передаче операнда по коммуникационной сети;

- регулирование параллелизма вычислительного процесса и управление распределением вычислений во времени и по пространству. К этому можно добавить и задачу мониторинга состояния ресурсов и интенсивности прохождения вычислительных процессов.

ППВС принципиально не имеет традиционной адресуемой памяти, с которой может работать пользователь в режимах записи и считывания.

Использование ассоциативной памяти (АП) позволяет извлекать из задачи даже неявный параллелизм, существующий в алгоритме ее решения. Вместе с этим ассоциативная память фактически выполняет многократные операции сравнения и проверки условия.

### B. *Исполнительное устройство*

Исполнительное устройство представляет собой процессор, основанный на фон-неймановской архитектуре. На вход ИУ поступает пакет с набором данных. Далее по адресу, считанному из пакета, выбирается из памяти команд программа, в соответствии с которой выполняется обработка данных. Результатом выполнения программы узла является генерация новых токенов. В настоящее время ведется проработка вопроса использования в качестве исполнительного устройства энергоэффективных процессоров фирм MIPS и ARM с расширенным набором команд и функциональных блоков.

### C. *Коммутатор токенов*

Коммутатор токенов распределяет токены по процессорам сопоставления. Токен, выходя из ИУ, поступает в блок хэш-функций, где на основе ключа токена происходит вычисление номера ПС, в который и направляется пришедший токен [4].

### D. *Коммутатор пакетов*

Коммутатор пакетов необходим для распределения готовых пакетов между свободными ИУ в ВМ. Коммутатор пакетов выполняет следующие функции:

- обеспечивает равномерную загрузку всех исполнительных устройств за счет направления готовых пакетов в наименее заполненное ИУ;

- обеспечивает связь всех ПС со всеми исполнительными устройствами ВМ. Использование подобно-

го коммутатора способствует сглаживанию имеющегося параллелизма задачи, так как каждый ПС генерирует готовые к выполнению пакеты неравномерно.

Коммутатор пакетов перекладывает распределение имеющихся аппаратных ресурсов (ИУ) с программиста на аппаратуру, фактически являясь аппаратным балансёром.

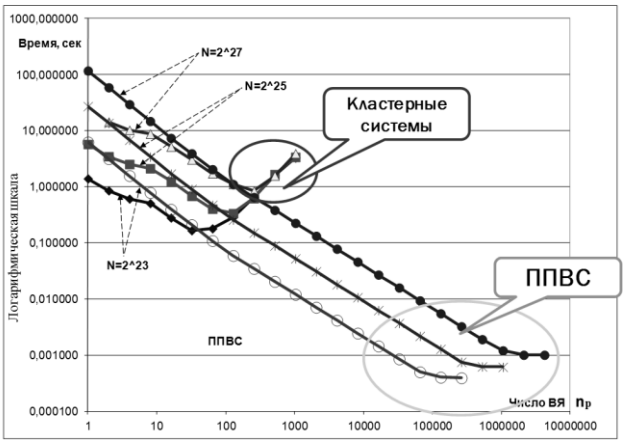


**Рис. 4. Сравнение возможности масштабирования вычислительных систем на задаче «Сортировка слиянием»**

## V. Предпосылки к созданию эксафлопсной системы

В последнее время за рубежом и в нашей стране специалисты пытаются сформулировать требования к архитектурным особенностям высокопроизводительных вычислительных систем эксафлопсной производительности. Большинство из них склоняются к тому, что для создания таких систем необходим «революционный» подход, использующий в полной мере новые парадигмы вычислений, новые параллельные языки программирования и неклассические архитектурные решения.

С архитектурной точки зрения (по мнению известного западного ученого профессора Университета Луизианы Томаса Стерлинга, создателя кластерной технологии Beowulf) системы эксафлопсной производительности должны обладать такими основными свойствами [4]:

- поддержка глобального адресного пространства, а не распределенной памяти;
- легкие механизмы синхронизации для быстрой передачи управления, а не принцип глобальных барьеров;
- динамические средства управления ресурсами, а не статическое распределение, выполненное при компиляции;
- обработка данных по мере их готовности, перемещая код к данным, в противоположность тому, как принято сейчас в моделях передачи сообщений;
- использование микроархитектуры, построенной на идеях обработки потока данных dataflow, а не на процессорных ядрах – так экономится энергия.

Архитектура ППВС «Буран» в той или иной степени поддерживает эти свойства, а также обладает и другими особенностями, позволяющими достигать высокой реальной производительности на широком круге актуальных задач.

## VI. Заключение

Были проведены исследования на поведенческой модели системы ППВС с использованием тестового пакета задач, в том числе на задаче «Сортировка слиянием», для которой на рис. 4 представлены результаты моделирования и экстраполяции данных. Эти данные демонстрируют преимущество в масштабировании ППВС по сравнению с классическими кластерными системами, которые строятся на основе зарубежных процессоров фирм Intel и AMD, практически на 4 порядка.

Таким образом, использование модели вычислений с управлением потоком данных, по нашему мнению, позволит решить большинство проблем при создании вычислительных систем эксафлопсного уровня. Архитектурная реализация этой модели вычислений в виде ППВС «Буран» обладает следующими возможностями:

- хорошее масштабирование системы, что позволяет создать многоядерный кристалл, а в дальнейшем и высокопроизводительные системы;
- аппаратное экстрагирование неявного параллелизма задачи в ходе ее решения;
- асинхронная работа отдельных блоков системы;
- нивелирование задержек в коммуникационной сети;
- аппаратно-программные технологии локализации вычислений по пространству и времени.

Эти и другие возможности модели вычислений и архитектуры ППВС позволят создать системы эксафлопсного уровня производительности.

## Литература


[1] http://www.zdnet.co.uk/news/processors/2011/06/22/tileras-100-core-processors-take-on-sandy-bridge-40093183/?s_cid=938&tag=mncol;txt (Дата обращения: 10.02.2012).

[2] http://www.intel.com/content/www/us/en/architecture-and-technology/many-integrated-core/intel-many-integrated-core-architecture.html (Дата обращения: 10.02.2012).

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

[4] Многоточие Стерлинга // Суперкомпьютер. №3. 2010. С. 17-20.