УДК 004.27

Способы регулирования вычислений в параллельной потоковой вычислительной системе

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

Институт проблем проектирования в микроэлектронике РАН (ИППМ РАН),
zmejevdn@ippm.ru, nick@ippm.ru, oku@ippm.ru, klimov@ippm.ru

***Аннотация* — В статье рассматриваются способы регулирования вычислительными процессами в параллельной потоковой вычислительной системе. Кратко описывается архитектура параллельной потоковой вычислительной системы и потоковая модель вычислений с динамически формируемым контекстом, которую она реализует.**

***Ключевые слова* — методы регулирования вычислительными процессами, модель вычислений с управлением потоком данных.**

I. Введение

В последнее время все большее число экспертов в области вычислительной техники как у нас, так и за рубежом, говорят о необходимости смены модели вычислений и, как следствие, архитектуры её реализующей. Это касается, прежде всего, высокопроизводительных вычислительных систем кластерного типа, которые демонстрируют низкую реальную производительность на актуальных задачах (на уровне нескольких процентов от их пиковой производительности).

Существующие на сегодняшний день решения по увеличению производительности одного процессора себя исчерпали. Остается один путь повышения производительности вычислительной системы – увеличение числа ядер в них. Одновременно с этим необходимо думать и о распараллеливании вычислительных процессов на большом количестве ядер для их эффективного использования.

Одним из путей решения данной проблемы является переход к новым оригинальным моделям вычислений. Для этого предлагается использовать потоковую модель вычислений с динамически формируемым контекстом, которая реализуется в архитектуре параллельной потоковой вычислительной системы (ППВС «Буран»).

II. Парадигмы «сбора» и «раздачи»

Важной особенностью потоковой модели вычислений с динамически формируемым контекстом является то, что программирование и, соответственно, работа системы осуществляются в парадигме «раздачи», в отличие от парадигмы «сбора», которая лежит в основе традиционных языков и систем (рис. 1). Суть парадигмы «раздачи» (рис. 1б) состоит в том, что инициатива в деле передачи данных принадлежит тому узлу, который породил данные, а не тому, кто в них нуждается. Более того, источнику данных не требуется (по крайней мере, логически) подтверждения факта получения данных. Это означает, что все передачи могут быть односторонними. Поэтому обмен токенами в потоковой модели — наиболее экономный способ взаимодействия с точки зрения количества передач и необходимых синхронизаций.

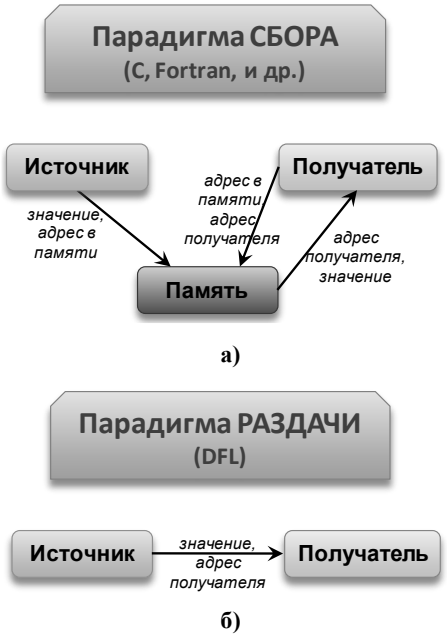

Рис. 1. Сравнение парадигм «сбора» (а) и «раздачи» (б)

При программировании на языке ППВС - DFL узел-источник сам «знает» (вычисляет) адреса всех потребителей (рис. 1б), на которые и рассылает свой результат. Это облегчает работу аппаратуры при распределенном выполнении.

В рамках парадигмы «сбора» (рис. 1а) источник сохраняет свой результат в памяти по некоторому адресу (рис. 1а), откуда его запрашивает потребитель по мере надобности.

Программировать для параллельной потоковой вычислительной системы можно, в принципе, как в парадигме «раздачи», так и в парадигме «сбора». Однако применение парадигмы «раздачи» значительно повышает эффективность работы вычислительной системы,

а также упрощает как процесс написания программ, так и создание компиляторов последовательных программ в параллельный язык ППВС – DFL.

Преимущества потоковой модели вычислений по сравнению с применяемыми в традиционных вычислительных системах приведены в работах [1, 2].

Эффективная работа многопроцессорных вычислительных систем невозможна без развитых средств управления вычислительными процессами.

Проведенные исследования продемонстрировали наличие ряда проблем, связанных с необходимостью введения средств управления вычислительными процессами. В частности, такими проблемами являются:

- необходимость регулирования параллелизма, как при его взрывном росте, так и при недостатке, когда имеющиеся аппаратные ресурсы простаивают из-за отсутствия вычислительных узлов, готовых к исполнению;

- необходимость балансировки загрузки вычислительных ядер системы;

- оптимизация работы коммуникационной сети, обмен в которой осуществляется «короткими» сообщениями;

- реализация «интеллектуального» ввода данных.

III. Особенности вычислительной системы «Буран»

Между ядрами в ППВС передаются единицы информации, которые называются токенами. Токен представляет собой структуру, содержащую данное (операнд), ключ, маску ключа, кратность и набор служебных полей. Как между ядрами в группе, так и между группами ядер действует один и тот же протокол взаимодействия, осуществляющий доставку токенов. Благодаря этому система может неограниченно масштабироваться.

Каждое ядро вычислительной системы (рис. 2) состоит из:

- процессора сопоставления (ПС), в котором происходит сопоставление токенов по определенным правилам;
- исполнительного устройства (ИУ), в котором происходит выполнение программы узла, которая выполняет обработку данных, содержащихся в пакетах;
- устройства хэширования, в котором на аппаратном уровне вычисляется функция распределения, определяющая номер целевого ядра токена.

Вычислительные ядра, в свою очередь, объединяются в вычислительный модуль, который может быть реализован в виде кристалла. В вычислительный модуль помимо перечисленных выше устройств входят:

- коммутатор пакетов, который передает пакеты, сформированные в процессорах сопоставления различных вычислительных ядер на любое свободное исполнительное устройство вычислительного модуля;
- внутренний коммутатор токенов, который пересылает токены, сформированные в исполнительных устройствах в соответствующие процессоры сопоставления, согласно номеру выработанному устройством хэширования.

В целом вычислительная система, которая реализует потоковую модель вычислений, обладает следующими особенностями, полезными для высокопроизводительных вычислений:

- имеется ассоциативная память с развитой систе-

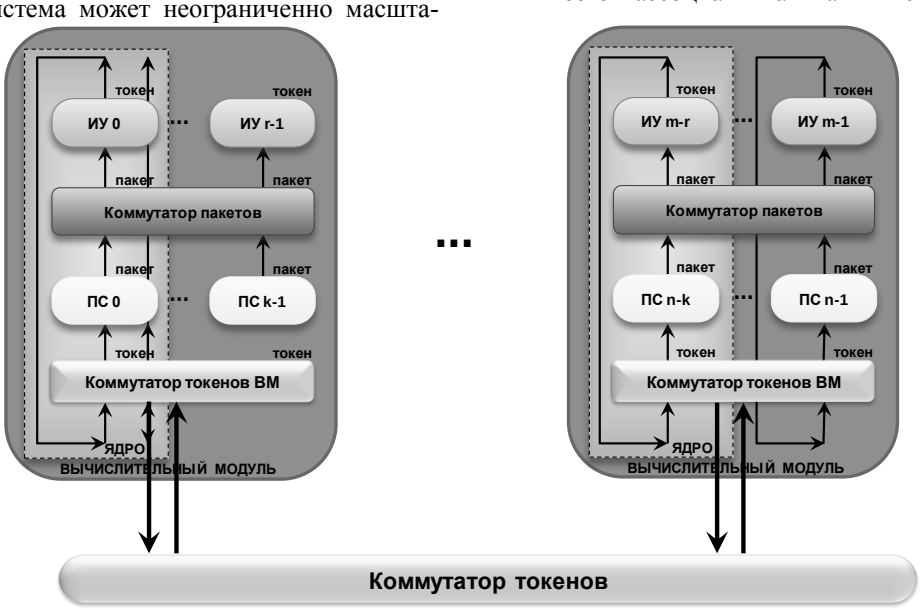

Рис. 2. Архитектура параллельной потоковой вычислительной системы

мой команд, на основе которой реализуется глобальное виртуальное адресное пространство (ключей токенов, или виртуальных узлов);

- система хорошо масштабируется, что позволяет реализовать возможность создания многоядерного кристалла, а в дальнейшем и высокопроизводительных систем на базе этих кристаллов;
- реализуется аппаратное выявление неявного параллелизма задачи в ходе ее решения;
- имеются аппаратно-программные средства управления параллелизмом задачи;
- имеет место асинхронность работы отдельных блоков системы;
- потоковая организация вычислительного процесса позволяет нивелировать задержки в коммуникационной сети.

IV. Варианты управления вычислительными процессами в классических dataflow системах

В классических dataflow системах, которые разрабатывались в конце 80-х, начале 90-х годов прошлого века, наибольшую известность получили следующие методы управления вычислительными процессами.

Метод “Loop pipelining”

Статический метод регулирования уровня параллелизма “Loop pipelining” [3] был разработан в Корнелльском и Калифорнийском университетах в США. Предназначался данный метод для использования на машине потока данных с явным хранением токенов Monsoon, а реализован был программно. Суть данного метода заключается в анализе компилятором программного кода и выявлении зависимости операций цикла или функции друг от друга. После такого анализа, компилятор разбивает операции по уровням.

Метод «throttling»

Динамический метод регулирования параллелизма “Throttling” был разработан создателями машины потока данных Manchester Data-Flow Machine [4]. В системе имеется устройство регулирования, которое собирает статистику по параллелизму, получая от всех процессорных элементов информацию об их загрузке.

Активации цикла или функции в системе предшествует запрос у устройства распределения ресурсов процессорным элементом номера активации. Этот номер используется в тегах токенов, принадлежащих этой итерации цикла или функции. Устройство регулирования анализирует уровень параллелизма системы. Если он достаточно высок, то на некоторое время может быть задержана выдача номера активации процессорного элемента. В результате приостанавливается активация функции или цикла, что снижает уровень параллелизма в системе.

Метод «K-bounding»

Статический метод регулирования параллелизма «K-bounding» разработан американскими специалистами, создателями MIT Tagged-Token Dataflow Machine в Массачусетском технологическом институте [5, 6]. Суть данного метода регулирования параллелизма заключается в ограничении числа одновременно выполняемых активаций цикла или функции до фиксированного числа k.

V. Аппаратно-программные методы управления вычислениями в ППВС «Буран»

Параллельная потоковая вычислительная система значительно отличается от классических систем dataflow (рис. 3), поэтому методы управления вычислительными процессами, разработанные для классических dataflow систем, напрямую не могут быть использованы в ППВС. Для неё были разработаны собственные методы управления вычислениями, только отчасти имеющие соответствие с классическими подходами.

Управление вычислительными процессами в ППВС можно разделить на программные и аппаратные средства. Эти средства могут работать как в статике, так и в динамике. Перечислим основные механизмы:

- распределение вычислений по пространству (локализация по вычислительным ядрам);
- распределение вычислений во времени (локализация по этапам);
- изменение алгоритма работы буферов токенов и пакетов;
- управление с помощью различных методов ввода данных в систему;
- ограничение числа активных временн***ы***х этапов (активаций) итераций или функций.

Наиболее универсальным методом управления вычислительными процессами в ППВС является локализация вычислений во времени с помощью задаваемых пользователем хеш-функций. Все токены вычислительного процесса разделяются хэш-функцией на некоторое количество групп, которые будем называть временн***ы***ми этапами (или просто этапами). Все этапы делятся на активные и пассивные. Имеется таблица этапов, куда заносятся номера всех незавершенных этапов с пометками об активности. По завершении работы этапа (когда в нем больше нет ожидающих токенов), этап исключается из таблицы этапов.

Активные этапы заполняют память процессора сопоставления, где происходит сопоставление токенов, принадлежащих соответствующему этапу.

Пассивные этапы находятся в памяти, где сопоставлений нет. Чтобы пассивный этап мог продолжить работу, он должен сначала стать активным.

Классические dataflow системы	Модель вычислений ППВС
Языки высокого уровня Sisal, Id, VAL и др. «работают» в парадигме «сбора».	Новая парадигма программирования – парадигма «раздачи», выраженная в языке DFL.
Заранее (статически) создается граф выполнения программы.	Граф выполнения программы создается (разворачивается) в процессе вычислений.
Процессора сопоставления не было (примитивные команды).	Имеется процессор сопоставления с развитой системой команд.
Не были решены проблемы управления локализацией и планирования вычислений.	Развитые (удобные) средства управления локализацией и планированием вычислений.
Были трудности с реализацией ассоциативной памяти ключей и токенов большого объема.	Благодаря реализации функций управления локализацией и планированием вычислений удалось избежать построения физической памяти большого объема (иерархия памяти).
Не была реализована поддержка динамического формирования контекста.	Реализована аппаратная поддержка динамического формирования контекста и функций взаимодействия (формирование токенов и пакетов).
Целевые узлы определялись в основном статически (были только элементы динамики).	Динамический выбор целевых узлов в узлах-источниках.

Рис. 3. Отличия ППВС «Буран» от классических dataflow систем

В отличие от классических систем 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.

Сам механизм ограничения количества одновременно исполняемых этапов может быть реализован несколькими способами. Во-первых, в «интеллектуальном» блоке ввода, во-вторых, непосредственно в процессоре сопоставления, в-третьих, частично он может быть реализован программными средствами на хост-машине. Первый и третий способы недостаточно универсальны, поскольку они воздействуют только на входные токены, позволяя вообще не вводить в коммуникационную сеть системы до поры «не нужные» токены. Токены, которые образуются в результате выполнения активных этапов, но сами относятся к пассивным этапам, в этом случае, будут оседать в процессоре сопоставления «мертвым грузом». Второй способ потребует большего объема оборудования для полноценной реализации на аппаратном уровне, но он намного более универсален.

VI. Заключение

В ППВС «Буран» управление вычислительными процессами осуществляется в блоке регулирования параллелизмом и через систему приоритетов токенов. Поскольку процессор сопоставления обладает развитой системой команд, которые позволяют изменять сам ход вычислительного процесса в динамике, то используя определенные команды и вводя новые можно легко расширять средства контроля и управления вычислениями в подобных системах.

Основным прикладным методом управления вычислительными процессами в ППВС «Буран» является локализация вычислений во времени с помощью задаваемых пользователем хеш-функций. Механизм ограничения количества одновременно исполняемых этапов какой-либо программы в системе может быть реализован разными способами. Используя возможности расширения системы команд ПС, средства поддерживающие установку приоритетов токенов и другие аппаратно-программные решения можно увеличить степень эффективности управления ходом вычислительного процесса в динамическом режиме.

Литература

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

[2] 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.

[3] Beck M., Pingali K.K., Nicolau A. Static scheduling for dynamic dataflow machines // Journal of Parallel and Distributed Computing. 1990. Vol. 10. P. 279-288.

[4] Gurd J., Böhm W., Teo Y.M. Performance Issues in Dataflow Machines // Future Generations Computer systems. Dec 1987. Vol. 3. № 4. P. 285-297.

[5] Arvind and R.S. Nikhil Executing a Program on the MIT Tagged-Token Dataflow Architecture // IEEE Trans. Comput. Mar. 1990. Vol. 39. № 3. P. 300-318.

[6] 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.