УДК 004.27

Исследование принципов работы блока ввода данных для параллельной потоковой вычислительной системы

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

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

***Аннотация* — В статье рассматривается влияние различных алгоритмов ввода данных на эффективность решения задач на ППВС "Буран". Оценивается, как выбор режима работы блока ввода данных связан с необходимым размером ассоциативной памяти ключей. Приводятся результаты экспериментов на программной модели ППВС.**

***Ключевые слова* — блок ввода данных, алгоритмы ввода данных, ассоциативная память.**

I. Введение

В последнее время эксперты в области высокопроизводительных вычислений говорят о кризисе параллельных вычислений, который проявляется в том, что растет число алгоритмов, эффективно распараллеливающихся не более чем на десятки, в лучшем случае на сотни ядер. Вследствие этого современные высокопроизводительные вычислительные системы кластерного типа, насчитывающие десятки и сотни тысяч ядер, на большом круге актуальных задач имеют реальную производительность в десятые доли процентов.

Одним из путей выхода из создавшегося положения является переход к новым моделям организации вычислений. Наши исследования позволяют сделать вывод о том, что для устранения проблем эффективного распараллеливания вычислений подходит потоковая модель вычислений с динамически формируемым контекстом. Данная модель вычислений реализуется в архитектуре параллельной потоковой вычислительной системы (ППВС «Буран») [1].

В параллельной потоковой вычислительной системе циркулируют данные – токены (рис. 1). Существует проблема перекодировки данных, поступающих извне (с хост-машины или внешних датчиков), и управления этими токенами для повышения эффективности работы системы. Данные проблемы призван решать блок ввода данных (БВД).

II. Параллельная потоковая вычислительная система

Модель вычислений, которую реализует архитектура параллельной потоковой вычислительной системы «Буран», основана на активации вычислительных квантов, которая происходит по готовности данных (рис. 2). Вычислительный квант представляет собой программу, которая, будучи активированной, выполняется до конца без привлечения дополнительной информации, то есть без приостановки процесса вычисления на подкачку дополнительных внешних данных. Различные вычислительные кванты между собой взаимодействуют и сохраняют состояние только через отправку сообщений, активирующих новые кванты. В узле-отправителе определяется и передаваемое значение, 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 [2].

Как было сказано выше, 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. Ключ, в свою очередь, состоит из полей «Адрес узла» и «Контекст» и однозначно идентифицирует место данного в виртуальном адресном пространстве задачи. Как между ядрами в группе, так и между группами ядер действует один и тот же протокол взаимодействия, осуществляющий доставку токенов. Благодаря этому вычислительная система может «неограниченно» масштабироваться.

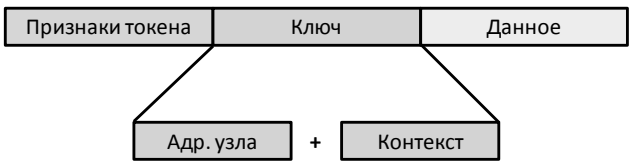


Рис. 1. Структура токена

В целях снижения семантического разрыва между программным и аппаратным обеспечением был разработан параллельный язык программирования ППВС «Буран» - DFL.

III. Основные функции и режимы работы блока ввода данных

Блок ввода данных поддерживает следующие режимы работы:

- «сквозной»;

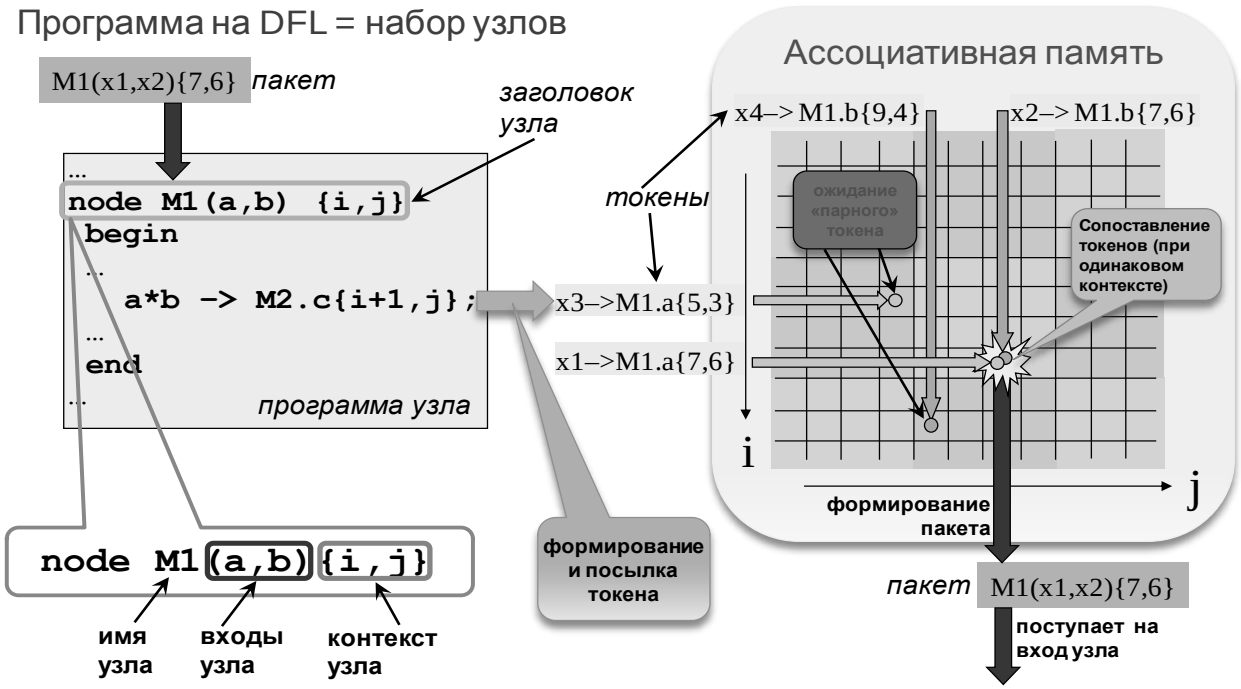


Рис. 2. Потоковая модель вычислений с динамически формируемым контекстом

- работа с шаблоном;
- работа со счетчиками;
- ввод разреженных данных;
- «порционный» ввод.

В режиме «сквозной» с хост-машины передается полностью сформированный токен, за исключением полей, определяемых на основании функций хэширования. Этот режим следует использовать для ввода в вычислительную систему одиночных токенов, а также данных нерегулярной структуры.

В режиме «работа с шаблоном» создается шаблон, в соответствии с которым токен будет формироваться и предварительно загружаться в блок ввода данных. Шаблон содержит поля расширенного ключа и признаков токена. Вместе с таким шаблоном допускается использование маски ввода данных пользователя, которая представляет собой битовую структуру, определяющую конкретные части полей формируемого токена, берущихся из шаблона или поступающих вместе с данными с хост-машины.

В режиме «работа со счетчиками» в блок ввода данных передаются маски счетчиков, определяющие размерность и положение индексов в поле ключа, а также значения верхних и нижних границ счетчиков. Этот режим необходим при передаче как одномерных, так и многомерных массивов, когда возникает необходимость включить в ключ индексы передаваемого элемента массива. В БВД происходит проверка (аппаратная) взаимного наложения масок. Счетчики имеют вложенность: внутренний – увеличивается с каждым следующим токеном в соответствии с заданной программой, а внешние – при достижении внутренними счетчиками «верхних» границ. Данный режим может использоваться совместно с предыдущим режимом работы.

В режиме «ввод разреженных данных», который может совместно использоваться с любым из имеющихся режимов, в вычислительную систему посылаются токены только с ненулевыми данными, что позволяет реализовать одну из особенностей параллельной потоковой вычислительной системы – отсутствие необходимости хранения «лишних» данных, над которыми не производится никакой работы. Возможно наложение на поле «данных» маски. Байты, закрытые маской, считаются нулевыми.

В режиме «порционный ввод», который также может быть использован совместно с другими режимами ввода данных, происходит управление вычислительным процессом. Например, при разбиении задачи на временны*е* этапы «активным» может быть только ограниченное количество этапов. В этом случае в вычислительную систему стоит отправлять только данные, которые относятся к активным этапам, а остальные этапы – задерживать в блоке ввода данных. Это позволяет значительно уменьшить объем ассоциативной памяти, требуемый для выполнения задачи. Также, изменяя размер «порции» данных, поступающей из БВД, можно регулировать параллелизм задачи. Одна из возможных реализаций заключается в подсчете количества отправленных и принятых токенов. После отправки заданного числа токенов БВД приостанавливает ввод до получения результатов их обработки, а затем отправляет следующую «порцию» токенов.

IV. Структура блока ввода данных

Блок ввода данных (БВД) конструктивно состоит из двух основных узлов: формирователя токенов (ФТ)

и узла ввода токенов (УВТ) (рис. 3).

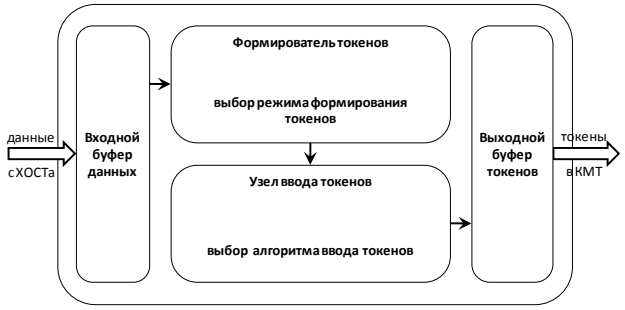


Рис. 3. Структурная схема блока ввода данных

Данные с хост-машины поступают во входной буфер данных БВД. Далее в формирователе токенов по заранее определенному режиму формирования токенов в соответствии с заданными шаблонами происходит формирование токенов – заполнение всех его полей.

На следующем этапе токены поступают в УВТ, в котором пользователь заранее выбирает хэш-функцию из уже имеющихся, определяет ее параметры или задает свою собственную хэш-функцию, вычисляя при этом номер подмножества токенов и номер вычислительного ядра. В этом же узле происходит анализ того, нужно ли задерживать токены в БВД или отправлять их в выходной буфер токенов для передачи в коммутатор токенов (КМТ) вычислительной системы.

V. Алгоритмы ввода токенов в КМТ

Для БВД были разработаны алгоритмы ввода данных, которые позволяют не только формировать токены из передаваемых данных, но и выстраивать очередность поступления токенов в вычислительную систему наиболее эффективным способом для каждой конкретной задачи. Перечислим основные режимы ввода данных:

- последовательный ввод по строкам;
- последовательный ввод по столбцам;
- поблочно-квадратный ввод;
- поблочно-квадратный половинчатый ввод;
- ввод колонками;
- ввод планками;
- блочно-точечный ввод;
- диагональный ввод;
- ввод векторов для БПФ.

На рис. 4 приведены графические представления некоторых алгоритмов ввода данных, для которых проводились исследования на поведенческой блочно-регистровой модели ППВС. Ввод данных (токенов) производится в порядке возрастания нумерации в направлении изображенных стрелок. Другими словами, номера, изображенные на рисунках, являются порядковым номером поступления входных токенов с данными.

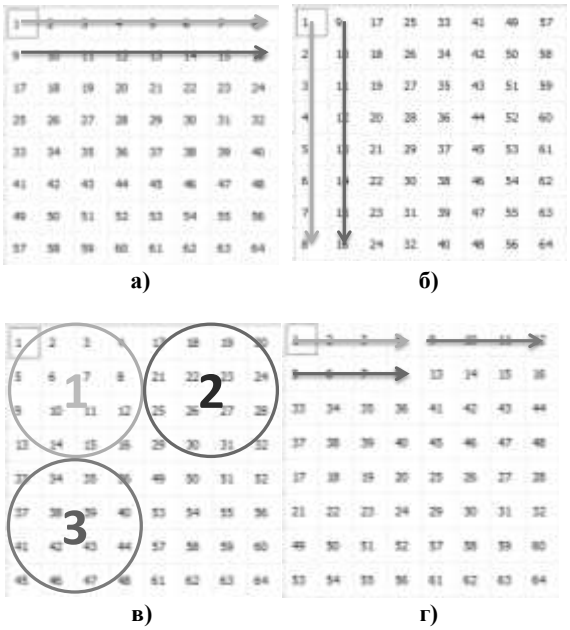


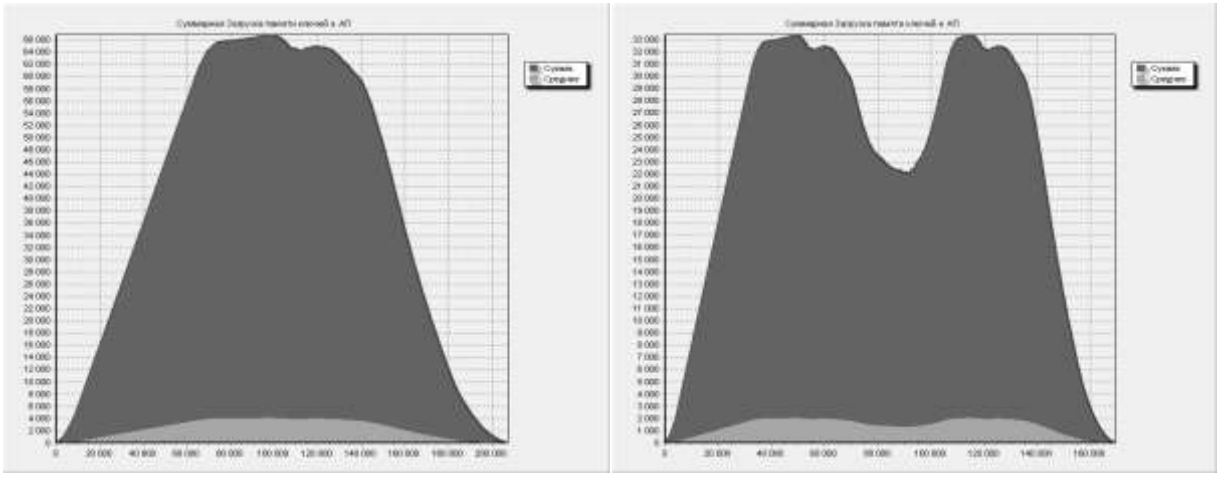
Рис. 4. Схемы алгоритмов ввода данных: а - последовательный ввод данных по строкам; б - последовательный ввод данных по столбцам; в - поблочно-квадратный ввод данных; г - поблочно-квадратный половинчатый ввод

На рис. 4а ввод данных осуществляется построчно в порядке возрастания номеров элементов массива. На рис. 4б ввод данных осуществляется по столбцам. На рис. 4в ввод данных осуществляется поблочно, каждый из блоков является квадратом. В приведенном примере матрица разбивается на 4 части, которые вводятся в систему последовательно по строкам внутри блока. На рис. 4г ввод данных осуществляется поблочно, но в отличие от предыдущего алгоритма, на первом этапе вводится верхняя половина блока, а на втором этапе, который начинается после ввода всех строк верхней половины блока, вводятся уже строки нижней половины блока данных.

VI. Исследование эффективности блока ввода данных на задачах

Исследование проводилось на задачах «Быстрое преобразование Фурье», «Сложение векторов», «Перемножение матриц» на программной блочно-регистровой модели параллельной потоковой вычислительной системы.

На рис. 5 приведена суммарная загрузка ассоциативной памяти ключей в устройстве сопоставления на задаче «Быстрое преобразование Фурье» для последовательного алгоритма ввода данных (рис. 5а) и алгоритма ввода данных методом чередования полувекторов (рис. 5б). Требуемый объем ассоциативной памяти ключей при применении второго метода ввода данных уменьшен в два раза (с 66000 ячеек ассоциативной памяти ключей до 33000). При этом время выполнения задачи уменьшилось



a) b)

Рис. 5. Суммарная загрузка памяти ключей на задаче «БПФ» при последовательном вводе данных векторов (а) и при вводе данных чередованием полувекторов (б)

приблизительно на 20% по сравнению с алгоритмом последовательного ввода данных.

На рис. 6 приведена суммарная загрузка ассоциативной памяти ключей на задаче «Перемножение матриц» при вводе двух матриц методом чередования. Видно, что максимальный объем ассоциативной памяти составляет всего 44 ячейки, что почти на два порядка лучше показателя по объему памяти ключей при последовательном вводе этих матриц (максимально необходимый объем ассоциативной памяти для этого метода ввода данных составил 3500 ячеек.).

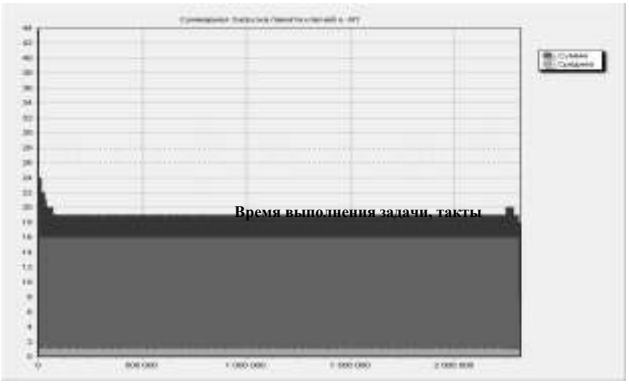


Рис. 6. Суммарная загрузка памяти ключей на задаче «Перемножение матриц» при вводе двух матриц методом чередования

VII. Заключение

В данной статье рассмотрены принципы построения блока ввода данных для параллельной потоковой вычислительной системы и особенности функционирования данного блока. Некоторые узлы БВД могут быть использованы для построения блока откачки/подкачки данных, являющегося частью иерархической ассоциативной памяти вычислительной системы.

Исследование влияния различных алгоритмов ввода данных на эффективность прохождения задач на ППВС и на необходимый размер ассоциативной памяти ключей процессора сопоставления связано также и с возможностью начала вычислений, не дожидаясь загрузки всего объема входных данных программы. Для этого и были разработаны различные алгоритмы ввода данных.

Анализ полученных в результате экспериментов данных показал, что суммарная загрузка ассоциативной памяти ключей в устройстве сопоставления на различных задачах при применении разных алгоритмов ввода существенно уменьшается. Кроме того, на задачах «Быстрое преобразование Фурье», «Перемножение матриц», «Сложение векторов» и других было получено ускорение прохождения задач в среднем на 20% по сравнению с прохождением этих задач без использования управляющих алгоритмов ввода данных.

Блок ввода данных играет важную роль в организации эффективной работы параллельной потоковой вычислительной системы.

Литература

[1] 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.

[2] 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.