Принципы функционирования вычислительной системы с набором команд дискретной математики DISC

А. Ю. Попов
Кафедра «Компьютерные системы и сети»
Московский государственный технический университет им. Н.Э. Баумана
Email:alexpopov@bmstu.ru

Аннотация. **Ускорение обработки больших объемов дискретной информации является приоритетной задачей для дальнейшего развития вычислительной техники. Для преодоления технологических ограничений современных вычислительных систем на выполнение операций над графами и множествами требуется разработка более совершенных принципов функционирования как аппаратного, так и программного обеспечений.**

В данной работе мы представляем принципиально новую вычислительную систему с многими потоками команд и одним потоком данных (МКОД), функционирующую на основе набора команд дискретной математики (Discrete mathematics Instruction Set Computer, DISC), которая была разработана и успешно функционирует в МГТУ им Н.Э. Баумана. Микропроцессор Leonhard, входящий в состав системы, позволяет хранить и независимо обрабатывать большие множества, структуры данных и графы. Мы приводим основные принципы функционирования вычислительной системы с набором команд DISC, а также перспективные области применения системы: роботизированные системы и программно-определяемые сети, системы машинного обучения и распознавания образов.

Ключевые слова: дискретная математика; графы; набор команд; структуры данных; множества

I. Введение

Рост объемов неструктурированных данных, непрерывно генерируемых устройствами Интернета вещей, социальными сетями и мобильными устройствами, делает чрезвычайно затруднительным их долговременное хранение и анализ универсальными вычислительными системами [1]. Перспективным направлением развития вычислительной технологии является создание гетерогенных систем в рамках проектов OpenPOWER [2] и HSA Foundation [3]. Однако, несмотря на открывающиеся перспективы, набор применяемых в гетерогенных системах ускорителей вычислений существенно ограничен устройствами обработки векторов и матриц, а также устройствами ускорения операций криптографии.

В этой работе мы рассматриваем вопросы аппаратного ускорения дискретной оптимизации, которая широко применяются для решения вычислительных задач в биоинформатике, экономике, статистике, телекоммуникациях, производстве, в торговом меркетинге, банковском секторе и других областях. Однако, до данной работы не существовало специальных средств, на аппаратном уровне поддерживающих весь набор операций дискретной математики. Важно также отметить, что современные универсальные вычислительные системы выполняют операции обработки множеств лишь посредством последовательности повторяющихся арифметико-логических операций [6], что не обеспечивает достаточного уровня быстродействия.

Цель этой статьи - представить принципы работы и перспективные направления внедрения вычислительной системы с несколькими потоками команд и одним потоком данных (MISD по классификации Флинна), которая была разработаны и функционирует в МГТУ им. Н.Э. Баумана. В составе такой системы параллельно функционируют два микропроцессора. Центральный процессор выполняет арифметические и логические команды, в то время как процессор обработки структур Leonhard (назван в честь Леонарда Эйлера) реализует набор команд дискретной математики (Discrete mathematics Instruction Set Computing, DISC).

II. Обзор предыдущих работ

Одним из ключевых в дискретной математике является понятие множества, которое в практике программирования принято реализовывать в виде структур данных, размещаемых в памяти ЭВМ. Для ускорения вычислений разработано большое количество различных типов структур данных, таких как массивы и деревья [4,5]. Однако, универсальные микропроцессоры и подсистема памяти вызывают существенные задержки доступа к информации для большинства ссылочных типов структур данных (деревьев, списков и пр.). Как отмечено в [4-7], существует значительный разрыв между временем доступа по последовательным и случайным адресам памяти. Это связано с тем, что доступ по случайным адресам вызывает гораздо больше промахов кэш-памяти, а также ошибок страничного доступа к виртуальной памяти по сравнению с последовательным доступом к таким структурам как вектора и матрицы. Кроме того, списки и деревья используют указатели для перехода от одного элемента к другому, что приводит к ситуации, когда адрес следующего элемента структуры становится известен

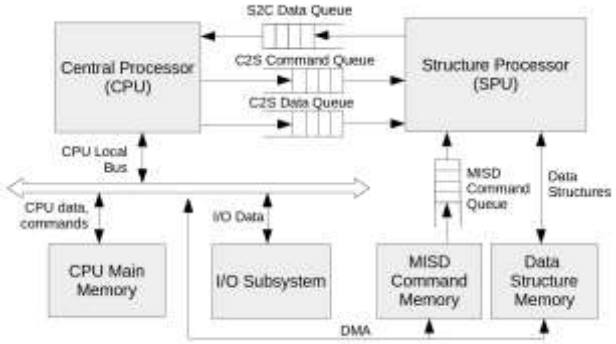

Рис. 1. Архитектура вычислительной системы с несколькими потоками команд и одним потоком данных

только после прохождения текущего указателя через весь конвейер процессора [9].

Результаты проведенных нами исследований новой архитектура и принципов функционирования системы с многими потоками команд и одним потоком данных были представлены в [7]. В работе отмечается, что набор команд процессора обработки структур соответствует операциям и кванторам дискретной математики. Приведенные оценки производительности системы позволяют перейти к ее внедрению в ряде важных применений. Определению наиболее перспективных направлений внедрения системы с набором команд DISC и посвящена данная работа.

III. Вычислительная система с поддержкой операций дискретной математики

Структуры данных с формальной точки зрения представляет собой совокупность двух сущностей: информационную составляющую о значениях полей данных (i) и структурную составляющую, учитывающую отношения данных (ii) [5]. Такая двойственность позволяет разделить процесс обработки структур данных на два потока вычислений: во-первых, поток обработки структурной составляющей и, во-вторых, поток обработки информационной части. В связи с этим на рис. 1 мы представляем архитектуру вычислителя, учитывающую эту особенность. Leonhard SPU - это специальный микропроцессор, который обрабатывает реляционную (структурную) часть структур данных, в то время как CPU выполняет вычисления над информационной составляющей структур [7].

У микропроцессора Leonhard имеется собственная независимая шина для доступа к локальной памяти, в которой происходит хранение структур данных. Результаты выполнения инструкций в SPU посылаются в CPU для дальнейшего их использования в вычислительном алгоритме.

В представленной системе существует два варианта запуска команды в SPU: команда запускается CPU, который отправляет ее через очередь команд C2S (i); SPU запускает команду, загруженную им из локальной памяти команд (ii). Если применяется первый вариант, то такая система является гетерогенной вычислительной системой (т.е. содержит неоднородные процессорные устройства). Если используется второй вариант, то такую систему можно классифицировать как параллельный компьютер с многими потоками команд и одним потоком данных [7].

Leonhard SPU использует два типа синхронизации выполняемых инструкций с потоком команд CPU. Первый способ использует семафоры для перевода блока выборки команд в состояние ожидания операндов. Для этой цели формат инструкций, которые поступают из локальной памяти команд (MISD Command Queue) включает в себя специальный бит тега для каждого из операндов (причем в одной команде может быть до трех операндов).

Во втором режиме CPU отправляет команды в SPU через очередь команд C2S только тогда, когда они действительно должны быть запущены и со всеми необходимыми операндами (аналогично взаимодействию математического сопроцессора с процессорам Intel 386). Таким образом, Leonhard SPU может работать в режиме MISD (первый вариант), в режиме ускорителя (второй вариант) и в комбинированном режиме (сочетаются оба способа синхронизации).

IV. Микроархитектура микропроцессора Leonhard

Микропроцессор Leonhard хранит информацию о множествах в виде неперекрывающихся B+ деревьев. Чтобы обеспечить быструю обработку информацию в дереве, SPU имеет конвейерную и параллельную микроархитектуру, показанную на рис. 2. SPU не только выполняет поиск в поддеревьях, операции вставки и удаления, но также выделяет и освобождает память для узлов и листов деревьев.

Мы разделяем процесс обработки B+ дерева на этапы «трассировки структуры дерева» и «обработку листьев». Это позволяет ускорить процесс вычисления по ключам и сохранить путь поиска во внутренней специализированной кэш памяти (которая носит название Catalog или CAT). CAT может выполнять обработку от корня дерева до листьев. Последовательность узла дерева на маршруте от корня дерева до его листа называется «трасса», элементы которой готовы к незамедлительному доступу в CAT.

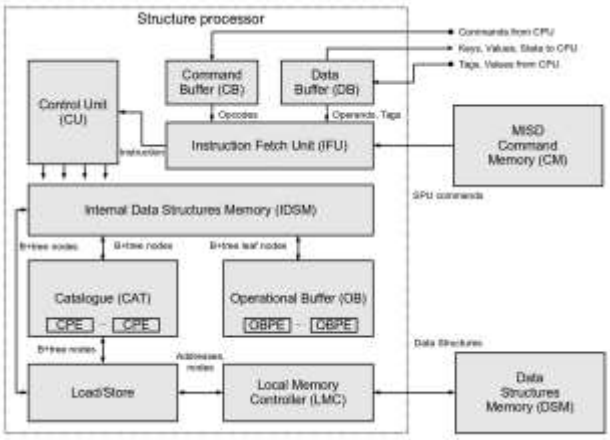

Рис. 2. Микроархитектура Leonhard

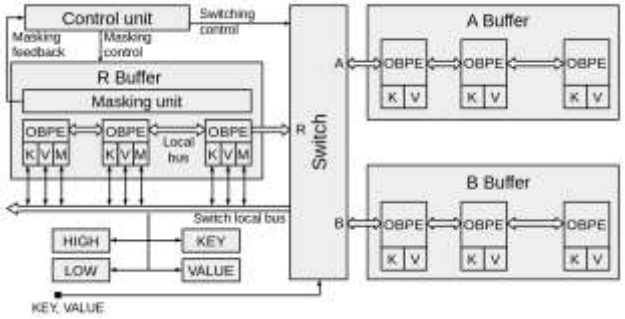

Рис. 3. Операционный буфер

После определения и загрузки трассы дерева в Operational Buffer (OB) начинается второй этап обработки, который используется для управления ключами и значениями на самом нижнем уровне. Подсистема памяти Leonhard включает в себя многоуровневые устройства хранения.

Первый уровень хранения представляет собой регистры внутри CAT и OB. Регистровая память организован как ассоциативный массив, что позволяет выполнять основные низкоуровневые операции: поиск, переключение, объединение и другие. Второй уровень - это внутренняя гранично-адресуемая память (Internal Data Structure Memory, IDSM), которая обрабатываются в CAT для определения физических адресов доступа к памяти следующего уровня. Третий уровень памяти – это внешняя оперативная память для долговременного хранения всей информации структур (Data Structures Memory, DSM).

Блок CAT содержит матрицу примитивных процессорных элементов (CPE) для обработки информации из нескольких поддеревьев. Блок управления (Control unit, CU) отправляет одну и ту же команду всем CPE для выполнения одной операции для разных данных. В связи с этим CAT имеет архитектуру SIMD, в то время как каждый CPE относится к классу SISD по классификации Флинна.

После того, как CAT определяет трассу B+ дерева, нужный лист загружается в OB из IDSM. Для поддержки таких операций, как объединение и пересечение, OB должен состоять из трех операционных блоков: Буферов A и B для хранения данных исходных структур, а также Буфера R для обработки и хранения структуры данных результатов (см. рис. 3). Все буферы в OB включают в себя примитивные процессорные элементы OBPE, что позволяет отнести архитектуру OB классу SIMD.

Полученный в Буфере R результат выгружается обратно в IDSM. Если требуется какая-либо дополнительная обработка, CAT снова загружает новую трассу и лист дерева в OB. В конце процесса, когда вся операция завершена, OB помещает результаты в очередь S2C.

V. Набор команд дискретной математики

Последняя версия набора команд Leonhard SPU была расширена двумя новыми инструкциями (NSM и NGR) для обеспечения требований некоторых алгоритмов. Каждая инструкция набора включает до трех операндов: ключа, значения и номера структуры данных. Набор команд состоит из 20 высокоуровневых кодов операций, перечисленных ниже.

Search (SRCH) выполняет поиск значения, связанного с ключом.

Insert (INS) вставляет пару ключ-значение в структуру. SPU обновляет значение, если указанный ключ уже находится в структуре.

Операция **Delete (DEL)** выполняет поиск указанного ключа и удаляет его из структуры данных.

Neighbors (NSM, NGR) выполняют поиск соседнего ключа, который меньше (или больше) заданного и возвращает его значение. Операции могут быть использованы для эвристических вычислений, где интерполяция данных используется вместо точных вычислений (например, кластеризация или агрегация).

Maximum /minimum (MAX, MIN) ищут первый или последний ключи в структуре данных.

Операция **Cardinality (CNT)** определяет количество ключей, хранящихся в структуре.

Команды **AND, OR, NOT** выполняют объединения, пересечения и дополнения в двух структурах данных.

Срезы (LS, GR, LSEQ, GREQ) извлекают подмножество одной структуры данных в другую.

Переход к следующему или предыдущему (NEXT, PREV) находят соседний (следующий или предыдущий) ключ в структуре данных относительно переданного ключа. В связи с тем, что исходный ключ должен обязательно присутствовать в структуре данных, операции NEXT/PREV отличаются от NSM/NGR.

Удаление структуры (DELS) очищает все ресурсы, используемые заданной структурой.

Команда **Squeeze (SQ)** дефрагментирует блоки памяти DSM, используемые структурой.

Команда **Jump (JT)** указывает SPU код ветвления, который должен быть синхронизирован с CPU (команда доступна только в режиме MISD).

Проведенные нами исследования в [7] показало зависимости по данным между потоками команд CPU и SPU для большинства алгоритмов оптимизации. В связи с этим мы предлагаем способ уменьшения зависимостей благодаря повторному использованию результатов, сохраненных во внутренних регистрах SPU. В большинстве случаев эти результаты могут быть повторно использованы как часть нового составного ключа (аналогично составным ключам в базах данных). Такой подход позволяет снизить зависимости по данным с 80-90% до 30-50% от всего потока инструкций SPU.

VI. Результаты экспериментов

Большинство команд SPU Leonhard требуют O (log n) операций доступа к памяти (исключая операции AND/OR/NOT и операции срезов), но они могут выполняться в несколько раз быстрее по сравнению с универсальными процессорами из-за эффективной параллельной обработки и кэширования трасс в SPU. Микропроцессор Leonhard был реализован на ПЛИС Virtex с встроенным микропроцессором PowerPC405. Были проведены эксперименты по измерению производительности полученной системы. Мы использовали первую версию Leonhard со следующими параметрами: 32-х битные ключи и значения; максимальное количество ключей в структуре $2 * 10^6$; максимальное количество структур данных: до 7; 256 МБ объем DSM; рабочая частота Leonhard SPU 100 МГц. Из-за значительной разницы частот между технологиями ПЛИС и СБИС мы измеряли количество тактов вместо времени выполнения (Таблица I). Это позволило нам понять преимущества новых принципов работы Leonhard без учета технологических факторов.

Пиковое потребление энергии для MISD системы, включающей микропроцессоры PowerPC и Leonhard, составило 1,1 Вт, что в 31 раз меньше 35 Вт при решении аналогичной задачи на микропроцессоре Core i5. Аппаратная сложность Leonhard составила 1,1 млн. эквивалентных вентилей, что вместе с 2,5 млн. вентилей микропроцессора PowerPC405 составляет в 1/420 аппаратной сложности 4-х ядер Core i5.

ТАБЛИЦА I Сравнение производительности Leonhard с универсальными микропроцессорами

Тестируемый алгоритм и микропроцессор	*Ускорение Leonhard*
Удаление (Microblaze)	164.4
Добавление (Microblaze)	42.7
Поиск (Microblaze)	31.4
Удаление (Intel Pentium 4)	22.8
Алгоритм Дейкстры (Intel Pentium 4)	19.4
Поиск (Intel Pentium 4)	15.3
Алгоритм поиска в глубину (ARM11)	12.9
Алгоритм поиска в ширину (ARM11)	12.3
Удаление (Intel Core i5)	11.8
Алгоритм Прима (ARM11)	10.3
Поиск (Intel Core i5)	9.8
Алгоритм Дейкстры (Intel Core i5)	7.6
Алгоритм Крускала (ARM11)	7.8
Добавление (Pentium 4)	5.7
Добавление (Intel Core i5)	3.2
Алгоритм поиска в глубину (Intel Core i5)	3.2
Алгоритм поиска в ширину (Intel Core i5)	3.0

VII. Внедрение системы

Мы рассматриваем несколько важных областей, в которых применение процессора Leonhard должно улучшить производитель и энергопотребление вычислительных систем.

В функции контроллера программно-определяемых сетей (SDN controller) входит сбор данных по протоколу OpenFlow от многих виртуальных или физических контроллеров сети. Большинство алгоритмов управления сетями строятся на основе графовых моделей и дискретной оптимизации. При этом ключевой характеристикой эффективности контроллера SDN является время его отклика и пропускная способность. С другой стороны, растут требования по аналитической обработке сетевых транзакций. Уже сейчас применяются средства машинного обучения для прогнозирования изменения траффика и предиктивного управления сетевыми ресурсами. Также требуется выполнять анализ траффика на предмет подозрительной активности приложений, сетевых атак и других угроз информационной безопасности. В связи с этим аппаратная поддержка SDN контроллеров на основе микропроцессора Leonhard является актуальной.

Другим направлением внедрения микропроцессора Leonhard являются робототехнические системы. Мо пере продвижения по маршруту (или заранее) робот строит граф видимости, в котором ребрами соединены видимые точки пространства. Вес ребра может отражать энергетические затраты, которые необходимы для перехода робота из одного положения в другое. В связи с этим требуется решать задачу поиска кратчайшего пути на графе видимости для эффективного перемещения робота.

Микропроцессор Leonhard был использован для алгоритма машинного обучения k-nearest и показал хорошие результаты производительности и точности на задаче классификации телеметрических данных программы Space Shuttle (достигнута точность распознавания 99.7%). Это позволяет использовать Leonhard в системах управления. В настоящее время развивается проект управления беспилотными летательными аппаратами и аппаратный автопилот. Также развивается проект применения Leonhard в системах компьютерного зрения, в которых необходимо не только распознавать видимые предметы, но управлять роботом на основе анализа сцены.

Список литературы

[1] Malik P. Governing Big Data: Principles and Practices, *IBM Journal of Research and Development*, 2013, vol. 57, no. 3a, 4, pp. 1:1-1:13.

[2] Stuecheli J., Blaner B., Johns C.R., Siegel M.S. CAPI: A Coherent Accelerator Processor Interface, *IBM Journal of Research and Development*, 2015, vol. 59, no. 1, pp. 7:1-7:7.

[3] Glossner J., Blinzer P., Takala J. HSA-enabled DSPs and accelerators, 2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP), Orlando, FL, 2015, pp.1407-1411.

[4] Knuth D. The Art of Computer Programming: Volume 1: Fundamental Algorithms, 3rd ed. *Addison-Wesley*, 1997, 672 p.

[5] Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms, 3rd ed. *MIT Press*, 2009, 1312 p.

[6] Tanenbaum A.S., Austin T. Structured computer organization, 6th ed. *Prentice Hall*, 2013, 801 p.

[7] Popov A. An introduction to the MISD technology, Proc. 50th Hawaii Int. Conf. on System Sciences (HICSS50), 2017, pp. 1003–1012.