ISSN 2079-3316 **ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ** **т. 9**, № 1(36), с. 37–52

УДК 004.27

Н. И. Дикарев, Б. М. Шабанов, А. С. Шмелёв

Моделирование параллельной работы ядер векторного потокового процессора с общей памятью

Аннотация. Процессор с архитектурой управления потоком данных может выполнять до 16 команд в такт по сравнению с 4–6 командами в такт у лучших процессоров фон-неймановской архитектуры. Моделирование векторного потокового процессора показало, что его производительность может быть доведена до 256 флоп в такт на ядро, и при изготовлении на кристалле с современными технологическими нормами можно разместить до 4 таких ядер.

Приводятся результаты моделирования системы из нескольких ядер векторного потокового процессора с общей памятью на программах перемножения матриц и решения систем дифференциальных уравнений 2D Stencil. Показано, что программа перемножения матриц масштабируется пропорционально числу ядер процессора, в то время как производительность 2D Stencil ограничивается пропускной способностью к общей оперативной памяти

Ключевые слова и фразы: суперкомпьютер, векторный процессор, архитектура управления потоком данных, оценка производительности, перемножение матриц, 2D Stancil.

Введение

В настоящее время все выпускаемые серийно процессоры имеют фон-неймановскую архитектуру, и их производительность близка к своему пределу, поскольку ни тактовая частота, ни число команд, выполняемых в такт, последнее время не растут. При этом ограничение пропускной способности конвейера команд у современных процессоров

Работа выполнена при поддержке гранта РФФИ 16-07-01124 и с использованием вычислительных ресурсов МСЦ РАН.

10.25209/2079-3316-2018-9-1-37-52

носит фундаментальный характер и определяется последовательной семантикой программ, свойственной фон-неймановской архитектуре процессора.

А именно, порядок выполнения команд в программе задается счетчиком команд, и часто команды должны выполняться последовательно. Например, если результат команды используется в качестве операнда следующей за ней командой, или выполняется команда перехода, определяющая следующее состояние счетчика команд. Для достижения пропускной способности в 4–6 команд в такт в современных процессорах используется сложная аппаратура управления, позволяющая изменять исходный порядок следования команд в программе, выдавая команды по готовности их операндов и осуществляя поиск таких команд в «окне» из примерно 200 команд. Квадратичный рост аппаратных затрат в этой аппаратуре при увеличении числа команд, выдаваемых в такт, приводит к снижению тактовой частоты и росту потребляемой мощности, что и определяет нецелесообразность дальнейшего роста производительности на скалярной обработке у процессорного ядра [1].

Поэтому увеличение производительности суперЭВМ осуществляется за счет роста числа процессорных ядер, которое в современных суперЭВМ достигает миллиона, что создаёт серьёзные трудности по эффективному использованию столь большого числа процессоров. Так для эффективного использования процессорных ядер необходимо, чтобы каждое ядро выполняло не менее 10 тысяч команд, поскольку при меньшем объёме вычислений процессор резко теряет производительность, в первую очередь, из-за недостаточного накопления данных в КЭШах. Кроме того, при увеличении числа процессоров увеличиваются потери времени на обмен данными и синхронизацию между ними, что также приводит к снижению производительности и при фиксированном объёме вычислений ограничивает возможность увеличения числа процессоров в суперЭВМ.

Очевидный способ решения данной проблемы, заключающийся в существенном увеличении производительности одного универсального процессора, в рамках фон-неймановской архитектуры, как показала практика, не может быть реализован. Однако мы считаем, что это можно осуществить за счет перехода на архитектуру управления потоком данных.

Процессор с архитектурой управления потоком данных (потоковый процессор) может обеспечить значительно более высокую производительность за счёт того, что параллелизм выполнения команд закладывается уже при составлении графа программы, и его не требуется выявлять с помощью сложной аппаратуры из последовательности команд, как это имеет место в фон-неймановском процессоре. Программой в потоковых ЭВМ является граф, узлами которого являются команды, а информация по дугам передается в виде токенов, содержащих значение операнда и номер команды приемника в графе программы. Причем вне зависимости от места двухвходовой команды в графе она выдаётся на выполнение по прибытию на вход последнего из пары токенов операндов.

После вычисления результата в исполнительном устройстве (ИУ) новые токены со значением результата отправляются на входы последующих команд согласно графу программы, а использованные токены операндов уничтожаются. Тем самым параллелизм выполняемых команд в потоковом процессоре определяется в динамике по мере прихода операндов на входы команд. При этом устройство поиска готовых к выполнению команд — память поиска пар готовых операндов (ППП) можно реализовать в виде многих работающих в параллель модулей, обеспечивающих работой такое же большое число ИУ. Таким образом, в потоковом процессоре можно сравнительно легко поднять темп выдачи команд до 16 команд в такт по сравнению с 4–6 командами в такт у лучших процессоров фон-неймановской архитектуры.

Цель данной работы — показать за счёт чего разрабатываемый в МСЦ РАН векторный потоковый процессор (ВПП) обеспечивает более высокую производительность по отношению к процессору фон-неймановской архитектуры, а также проанализировать его преимущества и недостатки при использовании в многопроцессорной системе. Кроме того, будут приведены первые результаты, полученные на модели из нескольких ядер ВПП с общей памятью по ускорению выполнения программ перемножения матриц и решения систем дифференциальных уравнений 2D Stencil.

1. Преимущества и недостатки ВПП при использовании в многопроцессорной системе

В настоящее время векторная обработка используется во всех высокопроизводительных процессорах, поскольку позволяет одной командой обрабатывать не одиночные данные, как при скалярной обработке, а одномерные массивы-вектора, увеличивая тем самым в несколько раз производительность процессора при той же пропускной способности конвейера команд [2]. В отличие от большинства современных процессоров, в которых используются короткие вектора (до 512 бит) с числом элементов не более 8 для чисел с плавающей запятой двойной точности, в ВПП аппаратная длина вектора VLmax=256 элементов по 8 байт.

Соответственно при обработке коротких векторов длиной 512 бит для достижения производительности 8 DP FLOP в такт нужно, чтобы 8 конвейерных ИУ сумматора или умножителя выполняли векторную команду с плавающей запятой каждый такт. В современных процессорах на векторной обработке вместо раздельных умножителей и сумматоров с плавающей запятой обычно используются «сдвоенные» умножители и сумматоры (Fused Multiply-Add (FMA)), выполняющие две операции с плавающей запятой в такт по вычислению функции $A \times B + C$, что позволяет вдвое повысить производительность при том же числе команд, выдаваемых в такт. При этом одно процессорное ядро в последних процессорах фирмы Intel Xeon Skylake и Knights Landing может выдавать до двух векторных FMA команд в такт, что при использовании расширения AVX512 к системе команд x86 даёт 32 DP FLOP в такт на ядро.

В ВПП также используются FMA ИУ, но за счёт увеличения длины вектора до 256 чисел с плавающей запятой двойной точности можно достичь значительно более высокой производительности. А именно, реализация векторного ИУ (ВИУ) в виде 32 идентичных колец (lanes) обработки позволяет обрабатывать вектора страницами по 32 элемента в такт. Тогда одно конвейерное FMA ИУ в каждом из 32 колец ВПП будет выполнять векторную команду длиной VLmax=256 элементов в течение 8 тактов, и один конвейер команд с пропускной способностью 1 команда в такт может загрузить работой до 8 FMA ИУ в каждом кольце.

Следовательно, даже один конвейер команд может обеспечить пиковую производительность ВПП, равную 512 флоп в такт, что в 16 раз выше по сравнению с лучшими процессорами Intel с двумя конвейерами команд с плавающей запятой в процессорном ядре. При использовании в многопроцессорных системах значительно более высокая производительность одного ядра ВПП по сравнению с процессорным ядром фон-неймановской архитектуры позволяет многократно снизить потери времени на обмен данными и синхронизацию между ядрами при одном и том же объёме вычислений.

Заметим, что векторные процессоры с обработкой длинных векторов — 256 элементов по 8 байт, как и в ВПП — выпускаются фирмой NEC. Процессорный кристалл NEC SX-ACE содержит 4 процессорных ядра, каждое из которых имеет производительность 64 флоп в такт, что обеспечивается параллельной работой 16 колец (lanes) обработки, по два сумматора и умножителя с плавающей запятой в кольце [3]. При этом пропускная способность к оперативной памяти у процессорного кристалла NEC SX-ACE равна 256 Гбайт в секунду — та же, что планировалась для процессорного кристалла ВПП и использовалась при его моделировании.

Таким образом, у NEC SX-ACE и ВПП много общего, и основное их различие в том, что процессорные ядра в NEC SX-ACE имеют фон-неймановскую архитектуру, а ВПП — архитектуру управления потоком данных. Как будет показано ниже, потоковая архитектура позволяет в несколько раз повысить производительность векторной обработки в ядре ВПП по сравнению с NEC SX-ACE за счёт более высокой степени векторизации программ, то есть той доли в общем объёме вычислений, которая выполняется с помощью векторных команд.

Другое несомненное преимущество ВПП, обеспечиваемое архитектурой управления потоком данных, заключается в его способности находить и выполнять команды с готовыми операндами в окне из примерно 20 тысяч команд по сравнению с 200 командами в лучших процессорах фон-неймановской архитектуры. Это даёт возможность выполнять в параллель команды не только из нескольких итераций внутреннего цикла программы, но и из итераций следующего вложенного цикла. В свою очередь, это позволяет не снижать производительность процессора при увеличении времени выборки из памяти до нескольких сотен тактов.

Аналогично могут быть компенсированы и задержки при обмене данными между процессорами в многопроцессорных вычислительных системах. Кроме того, выполнение программных блоков в ВПП начинается сразу по приходу отдельных значений входных операндов, а не после последнего операнда, что, в ряде случае, позволяет обойтись без глобальных барьеров.

Отметим ещё одно преимущество потоковых процессоров, обусловленное выполнением команд по готовности операндов и поиском таких команд в окне из нескольких десятков тысяч команд. Большое окно поиска готовых к выполнению команд позволяет одинаково успешно вести их поиск, как в графе одной программы, так и в графах нескольких программ.

Это означает, что процессор с архитектурой управления потоком данных может одновременно выполнять несколько программ, и в отличие от фон-неймановского процессора у него отсутствует специальная аппаратура, обеспечивающая переключение между потоками команд. Однако рост производительности можно получить, лишь если одновременно выполняемые программы задействуют разные типы ИУ процессора, а не те, что уже загружены одной из программ на 100%. Часто таким ИУ, ограничивающим производительность, оказывается тракт доступа процессора к оперативной памяти, а именно его недостаточная пропускная способность.

Несмотря на отмеченные выше преимущества потоковых процессоров, оказалось, что их трудно реализовать из-за ряда недостатков и сложностей практической реализации [4, 5]. Основные причины: сложность реализации ППП, которая должна обладать большой ёмкостью, поскольку её переполнение недопустимо, высоким быстродействием и осуществлять ассоциативный поиск. Вторая причина связана с использованием коммутатора, с помощью которого осуществляется пересылка токенов с выходов каждого из k ИУ на входы любого из k модулей ППП. Такой коммутатор должен иметь высокую пропускную способность по каждому из входов и одновременно малую задержку передачи до выхода, что трудно выполнимо уже при $k > 16$.

В разрабатываемом ВПП требования к ёмкости ППП существенно снижены, поскольку ППП не используется для хранения элементов массивов данных, они хранятся в двух уровнях обычной линейно-адресуемой памяти. Это память векторов (ПВ) — аналог оперативной

памяти и локальная память векторов (ЛПВ) — более быстрая и меньшей ёмкости память на процессрном кристалле. При этом ППП хранит лишь указатели массивов, преимущественно указатели векторов, которые в качестве одного из операндов векторной команды ожидают прихода второго операнда для выдачи этой команды на выполнение.

Описанное использование длинных векторов в ВПП позволяет сократить требуемую ёмкость ППП в сотни раз. При этом степень векторизации программ в ВПП выше, чем у векторных процессоров традиционной архитектуры за счет хранения массивов данных в ПВ и ЛПВ в виде «векторов-указателей», то есть, векторов, элементами которых являются указатели векторов подмассивов. Такой способ хранения массивов позволяет выделять свободное место, как в ПВ, так и ЛПВ, для записи результата каждой векторной команды. Тем самым гарантируется уникальный адрес вектора-результата векторной команды, необходимый для соблюдения принципа единственного присваивания в потоковом процессоре.

Распределение ресурса ПВ и ЛПВ в ВПП осуществляется на аппаратном уровне, и в качестве единицы фрагментации используется вектор с фиксированным числом слов VLmax=256. В этом случае входящее в состав ВПП устройство распределения памяти ведет список свободных векторов для ПВ и ЛПВ, и выделяет для записи результата векторной команды свободный вектор из затребованного списка.

Аппаратное распределение памяти в ВПП производится быстро, и снимает эту функцию с медленной операционной системы. Кроме того, такое хранение массивов в виде векторов-указателей позволяет одной командой «Формирование потока» (ФП) выдать токены с указателями, например, всех векторов строк из вектора-указателя матрицы для их последующей обработки в цикле. Тем самым удается векторизовать не один, а два вложенных цикла программы, и значительно сократить адресные и другие вычисления, выполняемые в скалярных ИУ (СИУ) ВПП, и соответственно повысить производительность ВИУ относительно СИУ [5]. В результате производительность одного ВИУ в ВПП можно поднять до 512 флоп в такт и уменьшить общее число скалярных и векторных ИУ, также как и модулей ППП до 8. Тогда в качестве коммутатора 8x8 можно использовать матричный коммутатор с малой задержкой, высокой пропускной способностью и небольшими аппаратными затратами.

2. Многопроцессорная система на основе ВПП и результаты её моделирования

Как уже отмечалось выше, производительность одного ядра ВПП на векторной обработке можно повысить как за счет увеличения числа колец векторной обработки, так и за счет увеличения числа FMA ИУ в кольце. Однако и в том и другом случае аппаратные затраты растут быстрее пиковой производительности из-за наличия коммутаторов. С увеличением числа FMA ИУ в кольце пропорционально растет число портов n в коммутаторе, подключающем входы и выходы FMA ИУ к банкам расслоенной ЛПВ, а затраты на коммутатор растут гораздо быстрее, почти как n^2. При увеличении числа колец векторной обработки в ВПП также быстро растут затраты на коммутатор в составе блока выполнения специальных операций (БВСО), поскольку с помощью коммутатора осуществляется доступ к ЛПВ в любом из колец.

БВСО может работать одновременно с векторным АЛУ, выполняя команды чтения и записи одиночных элементов вектора, команды сдвига, а также команды сборки-разбрасывания элементов вектора. Кроме того, с ростом пиковой производительности процессора его реальная производительность растёт не так быстро. Так на программе перемножения матриц у ВПП с 32 кольцами и 4 FMA ИУ в кольце производительность равна 243 флоп в такт или 95% от пиковой производительности в 256 флоп в такт. Моделирование той же программы при увеличении числа FMA ИУ в кольце до 8 показало, что отношение реальной производительности ВПП к пиковой уменьшается до 83%.

Быстрый рост аппаратных затрат при снижении отдачи в виде реальной производительности говорит о том, что не имеет смысла увеличивать производительность одного ядра ВПП сверх определённого предела. Поэтому было принято решение не доводить производительность одного ядра ВПП до 512 флоп в такт, и остановиться на 256 флоп в такт. В [6] было показано, что на программе перемножения матриц максимальная производительность с единицы площади кристалла достигается у ВПП при 32 кольцах, 4 FMA ИУ в кольце и расслоении ЛПВ на 16 двухпортовых банков. При 22 нм технологии изготовления СБИС этот экстремум составляет 2,19 флоп в такт с мм^2, что позволяет на кристалле площадью 450 мм2 разместить многопроцессорную

систему из 4 ядер ВПП с суммарной производительностью 1024 флоп в такт.

Для моделирования такой системы исходная VHDL модель уровня регистровых станций одного ядра ВПП была использована для построения модели многопроцессорной системы из нескольких ядер этого процессора с общей памятью, причём число ядер в модели можно задать в качестве параметра.

Каждое ядро ВПП в модели при обращении к общей памяти, которой является ПВ, получало доступ сразу ко всем каналам динамической памяти процессорного кристалла для доступа к ПВ через схему разрешения конфликтов. Меняющийся приоритет у процессорных ядер в этой схеме обеспечивал симметричный доступ к ПВ между всеми ядрами. Аналогичная схема разрешения конфликтов использовалась и для доступа к устройству распределения памяти, которое выделяет адрес свободного вектора для записи результата векторной команды, если он должен быть записан в ПВ, а не в ЛПВ.

Для распределения задач по процессорам был разработан граф управляющей программы, которая ведёт список свободных ядер в многопроцессорной системе и формирует токены с исходными данными для запуска очередного процесса на выполнение. Обрабатываемые процессом массивы данных должны находиться в ПВ, и управляющая программа посылает ядру токены с указателями этих массивов.

Разработанная VHDL модель многопроцессорной системы и управляющая программа отлаживались на программах перемножения матриц и 2D Stencil. Использовались блочные варианты этих программ, при которых обрабатываемые матрицы размером 512×512 состоят из 4 блоков размером 256×256, и в соответствии с описанным ранее методом хранения массивов хранятся в ПВ в виде трехмерного массива размером $4 \times 256 \times 256$. Одна и та же пользовательская программа, то есть перемножения матриц или 2D Stencil загружается в память команд каждого из процессорных ядер, а в память команд нулевого ядра — ещё и управляющая программа.

Программа перемножения матриц в процессорном ядре начинает вычисление подматрицы результата размером 256×256, получив от управляющей программы 3 токена, два из которых содержат вектора указатели матриц A и B размером 512×512, и третий токен передаёт номер вычисляемой подматрицы. В процессе вычисления по переданному номеру подматрицы результата программа перемножения матриц

вначале копирует из ПВ в ЛПВ ядра нужную подматрицу A размером 256×256 и затем 256 раз выполняет подпрограмму SGEMV [5]. После чего делается ещё один проход вычислений с копированием другой подматрицы A и суммированием с накопленными в первом проходе промежуточными результатами.

По окончании второго прохода вычисленная подматрица C переписывается из ЛПВ ядра в общую ПВ, и токен с вектором указателем вычисленной подматрицы отправляется управляющей программе в нулевое ядро. Управляющая программа записывает полученный указатель подматрицы в качестве элемента вектора указателя матрицы C и отмечает приславшее токен ядро как свободное, чтобы отправить ему задание на вычисление следующей подматрицы C, если такие остались.

Поскольку при вычислении подматрицы C в блочном варианте программы перемножения матриц элементы матрицы A читаются 256 раз, то её копирование из ПВ в ЛПВ позволяет в 256 раз уменьшить число обращений к ПВ, и ограниченная пропускная способность к ПВ не является узким местом, сдерживающим производительность на этой программе. Поэтому моделирование выполнения программы перемножения матриц на системе из двух ядер ВПП подтвердило высокую производительность работы каждого ядра при вычислении подматрицы C и пренебрежимо малые накладные расходы на переключение ядра с одной подматрицы на следующую.

Так перерыв в работе каждого ядра между вычислением первой и второй подматриц результата, обусловленный работой управляющей программы, составил меньше 200 тактов, в то время как ядро с производительностью 256 флоп в такт вычисляет подматрицу C за 279 тысяч тактов. В результате, время выполнения программы перемножения матриц размером 512×512 на системе из двух ядер ВПП составило 571 тысячу тактов, что соответствует производительности 478,7 флоп в такт или 93,5% от пиковой производительности. Если учесть, что блочный вариант той же программы на одном ядре выполняется с производительностью 243 флоп в такт, то ускорение составляет 1,97, что можно считать близким к линейному ускорению.

Что же касается выполнения программы решения систем дифференциальных уравнений 2D Stencil, то здесь ожидаемо было ограничение производительности из-за недостаточной пропускной способности к ПВ.

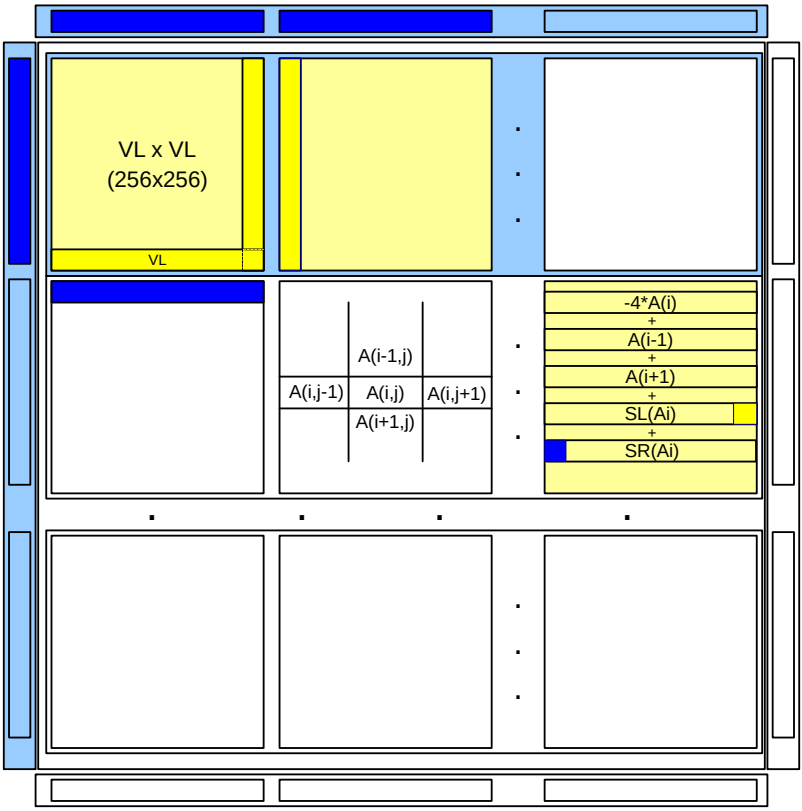

Рис. 1. Блочный алгоритм программы 2d stencil

Как показано на рис. 1, программа вычисляет новые значения элементов матрицы U из исходной матрицы A по следующему алгоритму: U(i,j)=A(i+1,j)+A(i-1,j)+A(i,j+1)+A(i,j-1)-4*A(i,j). После вычисления всех элементов матрицы U они используются для обновления матрицы A: A(i,j)=A(i,j)+ 0.1*U(i,j), и процесс повторяется Ntime раз. Вычисление матрицы U, как и затем новых значений элементов матрицы A, производится только векторными командами, с помощью которых каждая строка-вектор A(i) исходной матрицы умножается на константу -4.0, складывается со строками сверху A(i-1), снизу A(i+1) и ещё с двумя векторами. Это, как показано на рис. 1, результаты сдвига исходного вектора A(i) на один элемент влево SL(Ai) и вправо SR(Ai).

Поскольку при вычислении новой матрицы A на каждое исходное значение элемента этой матрицы приходится 7 операций с плавающей запятой, то копирование матрицы из ПВ в ЛПВ сокращает число обращений в ПВ лишь в 7 раз по сравнению с 256 раз в программе перемножения матриц, и мощное векторное АЛУ в ВПП оказывается не полностью загружено.

В блочном варианте этой программы управляющая программа помимо двух токенов с указателем матрицы A размером 512×512 и номером обрабатываемой подматрицы посылает в процессорное ядро еще указатели четырёх векторов, содержащие граничные элементы соседних подматриц, участвующих в вычислении новой подматрицы A (см. рис. 1). Поскольку управляющая программа передаёт ссылки на исходные данные, обрабатываемые программой, в виде токенов с указателями массивов в ПВ, то перед посылкой указателя столбца граничных данных управляющая программа для сборки элементов столбца читает по одному элементу в каждой строке из подматрицы справа и слева от подматрицы, подлежащей обработке.

Учитывая групповой обмен данными у динамических микросхем памяти, на которых построена ПВ, и работу лишь одного контроллера памяти из 16, на сборку элементов двух столбцов уходит то же время, что требуется на чтение всех 256 строк по 256 элементов, необходимых для копирования ядром подматрицы из ПВ в ЛПВ. То есть пропускная способность к ПВ при сборке элементов столбца в разработанном варианте графа программы 2D Stencil для многопроцессорной системы используется крайне неэффективно.

Поскольку управляющая программа при сборке элементов столбца в матрице, состоящей из векторов строк, тратит значительное время на обращения к ПВ, то пользовательская программа в нулевом ядре, также обращающаяся к ПВ в процессе работы, выполняется в 1,5 раза медленнее, чем в первом ядре. Соответственно при Ntime=10 из сорока подматриц 24 обрабатываются первым ядром и 16 — нулевым ядром. Что же касается однопроцессорного варианта той же программы, то здесь одновременно с обработкой текущей подматрицы осуществляется копирование следующей подматрицы из ПВ в ЛПВ для обработки следующего блока программы. В этом случае сборка элементов столбца для текущей подматрицы производится из ЛПВ, и обращений к ПВ не требуется. В результате производительность многопроцессорной системы из двух ядер ВПП оказалась лишь на 7% выше, чем у одного ядра ВПП.

Заключение

Результаты проведённого моделирования многопроцессорной системы из нескольких ядер ВПП подтвердили, что при увеличении числа ядер в системе можно получить близкое к линейному ускорение времени выполнения программы перемножения матриц, поскольку на этой программе пропускная способность к памяти не ограничивает рост производительности процессора. Моделирование также показало, что несколько программ, каких как блочное перемножение матриц и управляющая программа, могут выполняться на одном ядре ВПП с максимальной производительностью, так как они задействуют разные ИУ. Это позволило управляющей программе осуществлять переключение с выполнения одного блока перемножения матриц на следующий блок за время менее 200 тактов, что и обеспечило линейный рост производительности в многопроцессорной системе.

Моделирование времени выполнения программы 2D Stencil показало, что ВПП начинает выполнение блоков программы по мере прихода токенов с входными данными и значительную часть вычислений производит до прихода последнего токена с указателем столбца граничных данных блока. Эта способность процессора с архитектурой управления потоком данных позволяет поднять производительность ВПП при работе в составе многопроцессорной системы, также как и способность управляющей программы в динамике балансировать распределение нагрузки по ядрам.

Наконец, моделирование программы 2D Stencil подтвердило ещё одно преимущество работы процессорных ядер в составе многопроцессорной системы, а именно узкое место, сдерживающее производительность выполнения программы в однопроцессорной системе, перестаёт им быть при переходе к многопроцессорной системе. Так производительность выполнения программы 2D Stencil на одном ВПП ограничивалась пропускной способностью ИУ, выполняющего векторные команды сдвига.

Конечно, можно было за счёт дополнительных аппаратных затрат повысить производительность этого ВИУ, и такой вариант предлагался нами в [6], но оказалось, что при работе ВПП в многопроцессорной системе производительность программы ограничивает уже не производительность этого ВИУ, а пропускная способность процессорного кристалла к динамической памяти. Иными словами, для эффективной работы процессорного ядра в многопроцессорной системе требуется

более сбалансированный подбор производительностей разных типов ИУ, поскольку общую производительность ядер определяет такой фундаментальный фактор, как ограниченная пропускная способность процессорного кристалла к динамической памяти.

Проявился при моделировании программы 2D Stencil и недостаток разработанного варианта объединения нескольких ядер ВПП в многопроцессорную систему с общей памятью, заключающийся, как было показано выше, в увеличении числа обращений в память. Ограниченная полоса пропускания к ПВ позволила лишь незначительно поднять производительность выполнения этой программы при переходе от одного ядра к двум с той же полосой пропускания процессорного кристалла к ПВ. Способы преодоления этого недостатка, как программные, так и аппаратные, например введение общей для всех ядер кэш памяти на процессорном кристалле, будут анализироваться в процессе дальнейшей работы.

Список литературы

[1] Д. Л. Хеннеси, Д. А. Паттерсон. *Компьютерная архитектура. Количественный подход*, Пер. с англ., под ред. А. К. Кима, 5-е изд-е., Техносфера, М., 2016, 936 с., ISBN : 978-0-12-383872-8 ↑38

[2] Н. И. Дикарев, Б. М. Шабанов. *Архитектура высокопроизводительных вычислительных систем*, Фазис, М., 2015, 108 с. ↑40

[3] S. Momose, et al. “The brand-new vector supercomputer, SX-ACE”, ISC 2014: Supercomputing, Lecture Notes in Computer Science, vol. **8488**, pp. 199–214, ↑41

[4] G. V. Papadopoulos, K. R. Traub. “Multithreading: A revisionist view of dataflow architectures”, *Proc. 18-th Ann. Symp. on Computer Architecture*, 1991, pp. 342–351, ↑42

[5] Н. И. Дикарев, Б. М. Шабанов, А. С. Шмелёв. «Векторный потоковый процессор: оценка производительности», *Известия ЮФУ. Технические науки*, 2014, №12(161), Тематический выпуск: Суперкомпьютерные технологии, с. 36–46, ↑42,43,46

[6] Н. И. Дикарев, Б. М. Шабанов, А. С. Шмелёв. «Выбор оптимальной производительности ядра векторного потокового процессора», *Суперкомпьютерные технологии*, Материалы 4-й Всероссийской научно-технической конференции: в 2 т.. Т. 1, СКТ-2016 (19–24 сентября 2016 г.), Изд-во ЮФУ, Ростов-на-Дону, 2016, с. 36–41, ↑44,49

Рекомендовал к публикации *Программный комитет*
Шестого национального суперкомпьютерного форума ***НСКФ-2017***

Пример ссылки на эту публикацию:

Н. И. Дикарев, Б. М. Шабанов, А. С. Шмелёв. «Моделирование параллельной работы ядер векторного потокового процессора с общей памятью». *Программные системы: теория и приложения*, 2018, **9**:1(36), с. 37–52. doi 10.25209/2079-3316-2018-9-1-37-52

URL http://psta.psiras.ru//read/psta2018_1_37-52.pdf

Об авторах:

Николай Иванович Дикарев

Кандидат технических наук, ведущий научный сотрудник МСЦ РАН. Научные интересы: высокопризводительные вычислительные системы, архитектура управления потоком данных, векторные процессоры, параллельная и векторная обработка.

ID 0000-0002-7857-3250

e-mail: nic@jscc.ru

Борис Михайлович Шабанов

Кандидат технических наук, доцент, лауреат Государственной премии Российской Федерации в области науки и техники. Директор НИИСИ РАН.

e-mail: shabanov@jscc.ru

Александр Сергеевич Шмелёв

Научный сотрудник МСЦ РАН. Научные интересы: Суперкомпьютеры, архитектура управления потоком данных, векторные процессоры.

ID 0000-0002-1941-7792

e-mail: guest8993@rambler.ru

UDC 004.27

Nikolay Dikarev, Boris Shabanov, Aleksandr Shmelev. *Simulation of multicore vector dataflow processor with shared memory.*

Abstract. A Dataflow processor can execute up to 16 instructions per cycle compared to 4 to 6 instructions of the best von Neumann processors. Simulation of the vector dataflow processor (VDP) showed that it is possible to raise its core vector performance up to 256 flops per clock, and using modern manufacturing process to implement up to 4 such cores on a single die. Simulation results of the matrix multiplication program and 2D Stencil on double core VDP with shared memory are given in this paper. It is shown that the matrix multiplication program scales well on VDP, while the performance of 2D Stencil is limited by the shared memory bandwidth. (*In Russian*).

Key words and phrases: supercomputer, vector processor, dataflow architecture, performance evaluation, matrix multiplication, 2d stencil.

References

[1] J. L. Hennessy, D. A. Patterson. *Computer Architecture : A Quantitative Approach*, 5th ed., Morgan Kaufmann, 2011, 856 p., ISBN : 978-0128119051

[2] N. I.˜Dikarev, B. M.˜Shabanov. *Architecture of high performance computing systems*, Fazis, M., 2015 (in˜Russian), 108 p.

[3] S. Momose, et al. "The brand-new vector supercomputer, SX-ACE", ISC 2014: Supercomputing, Lecture Notes in Computer Science, vol. **8488**, pp. 199–214,

[4] G. V.˜Papadopoulos, K. R.˜Traub. "Multithreading: A revisionist view of dataflow architectures", *Proc. 18-th Ann. Symp. on Computer Architecture*, 1991, pp. 342–351

[5] N. I. Dikarev, B. M. Shabanov, A. S. Shmelëv. "Vector dataflow processor: performance evaluation", *Izvestiya SFedU. Engenreering Sciences*, 2014, no.12(161), Section I: Principles of construction, architecture and hardware base supercomputers, pp. 36–46 (in Russian)

[6] N. I.˜Dikarev, B. M.˜Shabanov, A. S.˜Shmelëv. "Choosing optimal performance of vector dataflow processor core", *Superkomp'yuternyye tekhnologii*, Materialy 4-y Vserossiyskoy nauchnotekhnicheskoy konferentsii SKT-2016: v˜2 t. V. 1 (19–24 sentyabrya 2016˜g.), Izd-vo YuFU, Rostov-na-Donu, 2016, pp. 36–41

Sample citation of this publication:

Nikolay Dikarev, Boris Shabanov, Aleksandr Shmelev. "Simulation of multicore vector dataflow processor with shared memory". *Program Systems: Theory and Applications*, 2018, **9**:1(36), pp. 37–52. (*In Russian*).

10.25209/2079-3316-2018-9-1-37-52

http://psta.psiras.ru//read/psta2018_1_37-52.pdf

10.25209/2079-3316-2018-9-1-37-52