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

Недостаточная пропускная способность памяти на программе Stencil: преимущество векторного потокового процессора

Аннотация. Основным фактором, ограничивающим производительность большинства приложений для высокопроизводительных вычислительных систем, является недостаточная пропускная способность памяти, а не недостаточная вычислительная мощность. Программными методами преодоления этого недостатка являются блочные методы, локализирующие обращения в пределах быстрой памяти на процессорном кристалле, и «программная конвейеризация» по организации вычислений в виде цепочек арифметических команд между командами обращения к памяти.

На примере программ 2D и 3D Stencil показано, что использование «программной конвейеризации» в разрабатываемом векторном процессоре с архитектурой управления потоком данных позволяет достичь значительно более высокой производительности по сравнению с лучшими процессорами традиционной архитектуры.

Ключевые слова и фразы: векторный процессор, архитектура управления потоком данных, многопроцессорная система с общей памятью, оценка производительности.

Введение

Для большинства приложений, решаемых на высокопроизводительных вычислительных системах в настоящее время, основным фактором, ограничивающим реальную производительность, является недостаточная пропускная способность памяти, а не недостаточная вычислительная мощность. Иными словами, большинство приложений являются memory-bound, а не compute-bound, хотя в 80-е годы прошлого века векторные процессоры, составлявшие основу парка суперЭВМ того времени, имели пропускную способность памяти 3 слова по 8 байт на 2 операции с плавающей запятой двойной точности, или 12 B/FLOP, что

Работа выполнена при поддержке грантов РФФИ 16-07-01124, 17-07-01457.

- © Н. И. Дикарев, Б. М. Шабанов, А. С. Шмелёв, 2018
- © Межведомственный суперкомпьютерный центр РАН, 2018
- © Программные системы: теория и приложения (дизайн), 2018

было достаточно для достижения максимальной производительности даже при отсутствии кэша данных. В 90-е годы, когда стало возможно разместить скалярные процессоры на одном кристалле интегральной схемы (ИС), эти векторные процессоры были вытеснены скалярными.

С ростом степени интеграции ИС производительность процессоров быстро росла как за счёт увеличения тактовой частоты, так и числа команд, выдаваемых в такт, а затем и за счёт роста числа процессорных ядер на кристалле. Пропускная способность памяти хотя и росла, но не так быстро как производительность процессоров, и разрыв между ними всё больше увеличивался. В результате отношение пропускной способности памяти к производительности у современных процессоров и ускорителей часто бывает меньше 0,1 В/FLOP. Единственный способ преодоления этого недостатка заключается в использовании нескольких уровней кэша, пропускная способность которых возрастает с уменьшением ёмкости, и самый быстрый из них — кэш первого уровня обеспечивает необходимые 6–12 В/FLOP. При этом для повышения производительности используются блочные алгоритмы, локализирующие обращения в пределах быстрой памяти на процессорном кристалле, и «программная конвейеризация» по организации вычислений в виде цепочек арифметических команд между командами обращения к памяти.

Программы решения систем дифференциальных уравнений в частных производных 2D и 3D Stencil [1] являются примером таких алгоритмов, в которых входные данные, перемещенные в локальную память процессора, обрабатываются цепочками из арифметических команд до получения выходных данных, записываемых обратно в оперативную память. Чем длиннее такие цепочки арифметических команд, тем меньше требуется пропускная способность памяти и, соответственно, значение В/FLOP. Однако в программах Stencil при вычислении каждого элемента выходных данных помимо значения этого элемента входного массива участвуют значения его соседних элементов. Их в современных процессорах до выполнения арифметических команд приходится читать из оперативной памяти в регистровый файл, задавая адреса этих элементов в памяти, и число команд обращения к памяти оказывается равным числу арифметических команд. Даже если обрабатываемый массив уместается в кэш второго уровня процессора, то его пропускная способность меньше требуемых 8 В/FLOP. При этом время выборки данных из памяти вместе со временем вычисления результата в конвейере исполнительного устройства (ИУ) с плавающей запятой составляет примерно 15 тактов. Тогда чтобы эти ИУ не простаивали

нужно одновременно выполнять команды из 15 итераций внутреннего цикла (цепочек арифметических команд), в то время как малая ёмкость регистрового файла не позволяет выполнять параллельно больше двух итераций цикла.

В результате производительность процессора на программах 2D и 3D Stencil составляет лишь 15% от пиковой производительности, и не меняется при увеличении длины цепочек арифметических команд. Экспериментальные данные, приведенные в [2] для процессора Intel Xeon Sandy Bridge, подтверждают это, как и то, что с увеличением размера обрабатываемых массивов производительность падает ещё в 8 раз, то есть до 2% от пиковой производительности процессора.

В разрабатываемом в МСЦ РАН векторном процессоре с архитектурой управления потоком данных, более кратко векторном потоковом процессоре (ВПП) порядок выполнения команд определяется исключительно готовностью их операндов. Недостатками процессоров этой архитектуры являются большее время выполнения приложений с низкой степенью параллелизма и высокие накладные расходы, связанные с передачей данных от одной команды к другой и поиском команд с готовыми операндами среди большого числа команд [3]. В ВПП эти недостатки в значительной степени были преодолены за счет введения векторной обработки и оригинального метода хранения массивов данных, позволившего повысить степень векторизации выполняемых программ [4]. Цель данной работы — показать, что разрабатываемый ВПП имеет значительно более высокую производительность выполнения программ Stencil, в том числе при увеличении размера обрабатываемых массивов, когда производительность процессоров традиционной архитектуры резко падает из-за недостаточной пропускной способности памяти.

1. Векторный потоковый процессор

Как и в большинстве современных процессоров, в ВПП для повышения производительности используется векторная обработка, что позволяет во много раз уменьшить число выполняемых команд, поскольку одна векторная команда заменяет собой цикл с независимыми итерациями с числом итераций VL , где VL — длина вектора. Основное отличие в том, что в ВПП используются длинные вектора с числом элементов равным 256, в то время как в современных процессорах — короткие вектора, в которых число элементов вектора — чисел с плавающей запятой двойной точностью — составляет от двух до

восьми. Обработка всех элементов вектора в современных процессорах осуществляется параллельно, для чего требуется от 2 до 8 конвейерных ИУ, и векторные команды к этим ИУ можно выдавать каждый такт. Поскольку высокопроизводительные процессоры способны выдавать до двух векторных арифметических команд в такт, то производительность процессорного ядра в этом случае увеличивается до 16 флоп в такт. Если же вместо отдельных команд сложения и умножения с плавающей запятой используются команды «сдвоенного» умножения и сложения (Fused Multiply-Add), которые вычисляют функцию $A*B+C$, то при выдаче двух таких FMA команд в такт производительность процессорного ядра возрастает до 32 флоп в такт.

В ВПП также используются FMA ИУ, но производительность значительно выше за счет обработки вектора страницами не по 8, а по 32 элемента в такт. При длине вектора 256 элементов 32 FMA ИУ в ВПП выполняют векторную команду в течение 8 тактов. Поэтому, выдавая даже одну векторную арифметическую команду в такт, можно загрузить работой до 8 комплектов по 32 FMA ИУ в каждом и довести производительность до 512 флоп в такт. Поскольку при увеличении пиковой производительности процессора его реальная производительность растёт не так быстро, то, как было показано в [5] на программе перемножения матриц, у ВПП максимальная производительность с единицы площади кристалла достигается при пиковой производительности не 512, а 256 флоп в такт. Поэтому было принято решение не доводить пиковую производительность ВПП до максимально возможной, а остановиться на производительности 256 флоп в такт. Соответственно результаты моделирования ВПП на программах 2D и 3D Stencil, приведённые ниже, получены на VHDL модели ВПП с производительностью 256 флоп в такт.

Еще одно существенное отличие ВПП от современных высокопроизводительных процессоров заключается в отсутствии регистровых файлов. В процессорах традиционной архитектуры конвейерные ИУ получают операнды из регистровых файлов и туда же записывают результат, а в разрабатываемом ВПП регистровые файлы не используются из-за их слишком малой ёмкости, которая обычно не превышает 32–64 слов. Такие же регистровые файлы используются в современных процессорах и при выполнении векторных команд. Просто каждое из 8 конвейерных ИУ, выполняющих векторную команду за 1 такт, подключено к своему регистровому файлу с номерами от 0 до 7 для обработки элементов вектора с теми же номерами. Аналогичное расслоение самой быстрой на процессорном кристалле памяти на модули

используется в ВПП. Только число модулей в этой памяти, которая называется локальная память векторов (ЛПВ), увеличено с 8 до 32, и к каждому модулю подключены не 2 конвейерных ФМА ИУ, а 4. При этом ёмкость каждого модуля ЛПВ не менее 24 К 64-разрядных слов, что в сотни раз больше чем ёмкость регистрового файла. В результате, суммарная ёмкость модулей ЛПВ составляет не менее 6 МВ, что позволяет хранить не менее 3000 векторов по 256 слов.

В ВПП требуется столь большая ёмкость ЛПВ, поскольку это процессор с архитектурой управления потоком данных. Поточковые процессоры могут обеспечить более высокую производительность по сравнению с процессорами традиционной архитектуры лишь, если в программе имеется достаточный уровень параллелизма, включая параллелизм уровня отдельных команд. Однако при выполнении цепочек из последовательных команд к задержке вычисления результата в ИУ у них добавляются значительные задержки в схеме управления, связанные с поиском готовых команд в ассоциативной памяти (АП) и передачей этих команд на выполнение в ИУ. Поэтому на чисто последовательном коде потоковые процессоры заведомо уступают в производительности процессорам фоннеймановской архитектуры, и компенсировать этот недостаток можно за счёт большего окна поиска готовых команд. Ёмкость АП в ВПП, которая и определяет размер этого окна, составляет 20 тысяч команд, в то время как в лучших современных процессорах — не более 200 команд. Это даёт возможность в ВПП осуществить параллельное выполнение команд из гнёзд циклов, в то время как в современных процессорах — лишь из нескольких итераций самого внутреннего цикла. Но для этого требуется быстрая память для хранения промежуточных результатов векторных команд ёмкостью не менее 3000 векторов [4]. Таким запоминающим устройством в ВПП, имеющим достаточную ёмкость и высокую пропускную способность для работы большого числа ИУ, не только на векторной, но и на скалярной обработке, является ЛПВ.

Помимо достаточно быстрой ЛПВ, расположенной на процессорном кристалле, в качестве основной памяти для хранения массивов в ВПП используется память векторов (ПВ) большой ёмкости, реализованная на микросхемах динамической памяти. Распределение адресного пространства ПВ и ЛПВ в отличие от вычислительных систем традиционной архитектуры в ВПП производится не программно, а аппаратно. Выделение свободного места в ПВ и ЛПВ производится для записи результата каждой векторной команды. Причем вне зависимости от числа обрабатываемых командой элементов вектора VL для записи

результата выделяется фрагмент, равный аппаратной длине вектора $VL_{\max}=256$ слов. Постоянная величина выделяемых фрагментов позволяет быстро и гибко распределять адресное пространство в ПВ и ЛПВ как между блоками одной программы, так и между программами, и свести к минимуму избыточный параллелизм [4]. Однако при такой фрагментации памяти для хранения массивов произвольной длины в ВПП приходится использовать древовидную структуру, когда вектор высокого уровня хранит в качестве элементов указатели векторов более низкого уровня.

При этом массивы, например матрицу размером больше чем 256×256 , приходится разбивать на подматрицы с числом элементов по каждому измерению не более $VL_{\max}$, и обрабатывать такие подматрицы последовательно. Это усложняет программирование, так как в программу нужно вводить дополнительные циклы, и увеличивает время доступа к произвольному элементу массива, но позволяет одной командой формирование потока (ФП) считать и выдать указатели векторов строк матрицы в качестве операндов для арифметических команд во всех итерациях цикла. Тем самым при выполнении цикла не требуется выполнять команды по вычислению адресов векторов строк матрицы, их чтению в регистровый файл из оперативной памяти, а также по подсчету числа выполненных итераций и выходу из цикла. В результате циклы с использованием ФП требуют для своего выполнения в 2–3 раза меньшее число команд по сравнению с фон-неймановским процессором [4]. Покажем это на примере выполнения программ 2D и 3D Stencil в ВПП.

2. Программы 2D и 3D Stencil и результаты их выполнения на Intel Xeon и ВПП

Программа решения систем дифференциальных уравнений 2D Stencil обновляет значения элементов матрицы A размером $(N+2) \times (N+2)$, не затрагивая элементы по периметру матрицы (граничные элементы). То есть пересчитываются элементы матрицы $A_{\text{матр}}$ размером $N \times N$ по следующему алгоритму:

Pseudocode –

```

1  DO k=1, N
2      U(k,1:N) =A(k+1,1:N)+A(k-1,1:N)+A(k,2:N+1)+A(k,0:N-1)
3  ENDDO
4  DO k=1, N
5      A(k,1:N) = 0,6*A(k,1:N) + 0,4*U(k,1:N)
6  ENDDO

```

Процесс пересчёта элементов матрицы Аматр по этому алгоритму повторяется N_{time} раз. Разбиение вычислений на два последовательных цикла, в первом из которых вычисляется матрица U размером $N \times N$ и лишь затем — матрица Аматр, говорит о том, что записывать новые значения элементов матрицы A можно лишь после того как все элементы исходной матрицы A были использованы в качестве операндов. Это нужно учитывать при реализации блочного варианта программы 2D Stencil, когда N превышает аппаратную длину вектора $VL_{\text{max}}=256$ в ВПП, но мы для упрощения графа программы сначала рассмотрим случай, когда размер N матрицы Аматр не превышает аппаратную длину вектора VL_{max} . В этом случае при вычислении каждой из N строк матрицы U нужно выполнить 3 векторные команды сложения и для обновления элементов строки Аматр — ещё две команды умножения и одну сложения.

Напомним, что программой для потокового процессора, и ВПП в частности, является граф, в котором узлами являются команды, а дуги служат для передачи результата с выхода команды на входы других команд, где этот результат будет использован в качестве операнда. Данные по дугам графа передаются в составе пакета — токена, который помимо поля данных содержит ещё несколько полей. У токена в ВПП это поле с номером команды приёмника K , а также для параллельного выполнения команд в циклах и подпрограммах — поля I и T с номерами итераций двух вложенных циклов и поле P с номером подпрограммы.

Команды выдаются из АП на выполнение в ИУ по приходу последнего токена операнда с одинаковыми полями K , P , T и I , что позволяет выполнять команды по готовности операндов. Токены операнды после выполнения команды в ИУ уничтожаются, только для векторных команд уничтожаются лишь указатели векторов в поле данных токена, но не значения элементов вектора в ПВ или ЛПВ. Адреса векторов после их использования в качестве операнда в последний раз возвращается в список свободных векторов в устройстве распределения памяти ВПП. Для этого вход векторной команды помечается признаком уничтожения, либо токен с указателем вектора посылается на вход команды уничтожить вектор (VB).

Граф для выполнения в ВПП программы 2D Stencil, показанный на рис. 1, содержит внешний цикл с числом итераций N_{time} . Константа N_{time} приходит в составе токена на вход команды 1 в графе при пуске программы. Вектор указатель обрабатываемой матрицы Аматр,

РИСУНОК 1. Граф программы 2D Stencil

элементами которого являются указатели векторов строк матрицы $A(k,1:N)$ (для k от 1 до N), поступает из внешней машины на вход команды дублирования ПДБ 16. Аналогично при пуске программы на входы команд 18, 20, 22, 21 приходят токены с указателями четырех векторов, содержащих элементы нулевой и $(N+1)$ строк, а также нулевого и $(N+1)$ столбцов исходной матрицы A , участвующие в вычислении новых значений её элементов.

Заметим, что в отличие от традиционного хранения матрицы произвольного размера в памяти ЭВМ одним массивом, в ВПП матрица A при $N = 256$, то есть размером 258×258 хранится в ПВ как 256 векторов строк по 256 элементов, плюс её вектор указатель Аматр плюс 4 вектора граничных условий. В графе не показаны команды, которые копируют все эти вектора до начала вычислений из ПВ в ЛПВ. Это необходимо для локализации обращений к памяти при выполнении программы в пределах ЛПВ, и позволяет вычислять новые значения элементов матрицы Ntime раз без обращений к ПВ. Лишь по окончании вычислений результирующую матрицу Аматр нужно будет переписать в ПВ с одновременным уничтожением векторов

в ЛППВ, то есть с освобождением ресурса локальной памяти.

По приходу вектора указателя матрицы Аматр на вход графа токен с указателем этого вектора запускает команду дублирования 16. Один из токенов с её выхода поступает на вход команды ФП1 с номером 6 в графе на рис. 1. Как и другие команды с одним входом команда ФП1 сразу выдаётся на выполнение. Она читает в ЛППВ адреса векторов строк матрицы Аматр, формирует из них 256 токенов для команды векторный сдвиг вправо ВСДП 38 и ещё один токен с указателем матрицы Аматр передаёт по второму выходу на команду дублирования 14. Тем самым команды дублирования и ФП1 размножают токен с указателем матрицы Аматр необходимое число раз, чтобы запустить на выполнение ещё несколько команд ФП и сдвига. Так команды ФП 16 и 29 читают в памяти и выдают на векторную команду сложения 32 указатели векторов строк, на одну выше и ниже вычисляемой строки. Это достигается сдвигом элементов вектора указателя Аматр на один элемент вправо командой СДВП 26 и влево командой СДВЛ 27 с установкой на место освободившегося элемента указателей векторов нулевой и $(N + 1)$ строк исходной матрицы A , то есть граничных условий.

Аналогично с помощью команд сдвига СДВП 31 и СДВЛ 12 в графе на рис. 1 формируются вектора, сдвинутые на один элемент вправо и влево относительно вычисляемой строки, которые поступают входы векторных команд сложения 28 и 33 соответственно. Далее вычисленные командой 33 (сложения с плавающей запятой) вектора строки матрицы U поступают для дальнейшего умножения командой 34 на константу 0,4. Затем в соответствии с приведённым выше алгоритмом команда сложения 37 вычисляет новые значения векторов строк матрицы A , которые записываются по адресу первого операнда команды сложения 37, то есть по адресам векторов, содержащимся в векторе указателе Аматр.

На этом процесс вычислений в цепочке векторных арифметических команд, начиная с команды сложения 32 до команды сложения 37, заканчивается. Если это не последняя итерация по $Ntime$, то команды «конец цикла по итерации» КЦТ 23 и 30, 33, 36, 38 возвращают вектор указатель Аматр и указатели 4 векторов граничных условий соответственно, на исходные позиции в графе, начиная тем самым следующую итерацию по $Ntime$.

Заметим, что в графе программы 2D Stencil, показанном на рис. 1, вычисление новых значений элементов матрицы Аматр производится

только векторными командами, которые на рис. 1 выделены более тёмным цветом. Причем последовательности токенов из векторов операндов для арифметических команд, формируемые командами ФП, содержат адреса этих векторов в ЛПВ в явном виде. Достаточная для хранения нескольких матриц, размером 256×256 слов, ёмкость ЛПВ позволяет обойтись в ВПП без команд обращения к памяти в процессе вычислений. В процессоре традиционной архитектуры такие команды необходимы не только для подкачки порциями векторов из оперативной памяти в регистровый файл, но и для чтения векторов со сдвигом относительно исходного вектора. В итоге производительность выполнения этой программы в современных процессорах ограничивается как малой ёмкостью регистрового файла, так и недостаточной пропускной способностью конвейера команд (не более 6 команд в такт).

Как уже отмечалось выше, эти факторы, вместе с задержкой выборки данных из памяти в 10 тактов, не позволяют выполнять параллельно больше двух итераций внутреннего цикла, и производительность современных процессоров на программе 2D Stencil составляет не более 15% от их пиковой производительности. В ВПП задержка выборки данных из ЛПВ также считалась равной 10 тактам при моделировании. Однако токены векторов операндов для арифметических команд, формируемые командами ФП, имеют разные значения номера итерации k внутреннего цикла, записанные в поле И токена. Это позволяет ВПП выполнять одновременно команды из нескольких сотен итераций цикла, и АП имеет достаточную для этого ёмкость. Тем самым ВПП может не снижать производительность при увеличении времени выборки из памяти до 100 тактов.

Система авторегулирования нагрузки в ВПП раскручивает параллелизм до уровня, при котором начинает образовываться очередь из готовых команд к наиболее медленному ИУ. На данной программе таким ИУ в ВПП является векторное ИУ, выполняющее команды сдвига СДВП и СДВЛ. Это конвейерное ИУ, также как и векторное АЛУ, обрабатывает вектора страницами по 32 элемента в такт, и его максимальная пропускная способность при $VL=256$ равна одной команде раз в 8 тактов. Поскольку пропускная способность векторного АЛУ в ВПП в 4 раза выше, и на две команды сдвига (СДВП 31 и СДВЛ 12 в графе на рис. 1) приходится 6 векторных арифметических команд, то максимально возможная производительность ВПП на этой программе не может быть превышать $\frac{3}{4}$ от пиковой производительности. На данной программе ФМА команды не использовались, и пиковая

производительность ВПП в этом случае равна 128 флоп в такт, а максимально возможная — 96 флоп в такт.

Результаты моделирования подтвердили, что в ВПП время выполнения программы 2D 5-pt Stencil (5 точек сетки участвуют в вычислении одного элемента) определяется временем выполнения команд сдвига, доля которого составляет 94% от общего времени выполнения программы, а производительность равна 90,2 флоп в такт. Это максимально возможная производительность ВПП на этой программе, и достигается, как отмечалось выше, лишь при размере обновляемых элементов сетки 256×256 . При увеличении размера сетки нужно использовать блочный вариант этой программы, который отличается от рассмотренного разбиением большого массива на подматрицы с числом элементов по каждому измерению не более V_{Lmax} и обработкой таких подматриц последовательно. Такой алгоритм был реализован, причем во время вычисления новых значений элементов одной подматрицы подгружались из ПВ в ЛПВ вектора строки следующей подматрицы, нулевые элементы которых использовались в качестве столбца граничных условий [6].

Производительность блочного варианта программы 2D 5-pt Stencil была ниже из-за увеличения числа векторных арифметических команд, с помощью которых копировались строки следующей подматрицы. Здесь использовалась возможность ВПП при выполнении векторных команд читать вектора операнды как из ЛПВ, так и из ПВ, и также записывать результат в память любого уровня. Как было показано в [6], одна итерация внешнего цикла (по Ntime) по вычислению новых значений матрицы A размером 512×512 на одном процессорном ядре ВПП выполняется за 24,1 тысячи тактов, что соответствует производительности 65,3 флоп в такт.

Эта производительность в 1,4 раза ниже, чем при размере матрицы 256×256 (90,2 флоп в такт). Тем не менее, производительность ВПП на матрице размером 512×512 в 4,5 раза выше, чем при выполнении на двух шестиядерных процессорах Intel Xeon X5680 (Sandy Bridge) с частотой 3,3 ГГц на том же размере матрицы [2]. Заметим, что система на процессорах Intel Xeon X5680 на этом размере матрицы (512×512 или 2 МБ) достигает своей максимальной производительности 14,55 флоп в такт за счёт локализации вычислений в кэш второго уровня.

Если же учесть, что с увеличением размера матрицы в этой программе до 32 МБ производительность системы на Intel Xeon

падает в 8 раз, то преимущество ВПП станет ещё больше. Дело в том, что при увеличении размера матрицы в блочном варианте этой программы производительность ВПП останется неизменной, так как программа на каждом проходе по Ntime копирует очередную подматрицу из ПВ в ЛПВ и вычисленную подматрицу записывает обратно в ПВ. Таким образом, при увеличении размера матрицы до 32 МБ разрыв в производительности между ВПП и системой из 12 ядер Intel Xeon вырастет до 35 раз. Конечно, более справедливая оценка должна учитывать пропускную способность процессорного кристалла к динамической памяти, которая и определяет производительность программ Stencil, когда размер обрабатываемого массива не умещается в кэш или в ЛПВ применительно к ВПП. Согласно [2], пропускная способность двухпроцессорной системы (12 ядер) на Intel Xeon X5680 составляет 20 GB/s, в то время как у ВПП при моделировании считалась равной 256 GB/s. Тогда при размере матрицы больше 32 МБ производительность выполнения программы 2D 5-pt Stencil на единицу пропускной способности памяти у ВПП окажется в 2,8 раза выше по сравнению с системой на Intel Xeon.

Меньшее преимущество ВПП при сравнении по удельной производительности на единицу пропускной способности памяти объясняется разным соотношением пропускной способности памяти и пиковой производительности. Так у системы из 12 ядер Intel Xeon X5680 пиковая производительность на частоте 3,3 ГГц равна 96 Гфлопс, и отношение пропускной способности к пиковой производительности составляет 0,21 B/FLOP. При этом то же отношение у ВПП составляет 1 B/FLOP, если FMA команды используются и 2 B/FLOP, если FMA команды не используются, то есть удельная пропускная способность памяти у ВПП почти в 10 раз больше.

На рис. 2 приведено сравнение производительности ВПП с вычислительным узлом на процессорах Intel Xeon X5680 для случая, когда влияние пропускной способности памяти на производительность выполнения программ Stencil несущественно. Для системы из 12 ядер Intel Xeon такие условия, обеспечивающие максимальную производительность, создаются при размере массива 2 МБ, когда массив умещается в кэш второго уровня, и при проходах по Ntime обращений к более ёмкой памяти не требуется. Что же касается ВПП, то это блочный вариант выполнения программы, когда копирование очередной подматрицы из ПВ в ЛПВ производится одновременно с обработкой текущей подматрицы, и время обработки подматрицы определяется,

РИСУНОК 2. Сравнение производительностей ВПП и Xeon на программах Stencil

в первую очередь, временем выполнения команд векторного сдвига, а не временем копирования.

Как видно из зависимостей, приведенных на рис. 2, при увеличении числа точек (элементов), участвующих в вычислении новых значений элемента массива в программе Stencil, и соответственно операций с плавающей запятой на одну итерацию внутреннего цикла, производительность ВПП увеличивается, а системы из 12 ядер Intel Xeon — нет. У системы на Intel Xeon это объясняется пропорциональным увеличением числа команд обращений к памяти с ростом числа операций с плавающей запятой при вычислении каждого элемента. Что же касается ВПП, то здесь увеличение числа операций с плавающей запятой происходит при одном и том же числе обращений к памяти и операций сдвига вектора. Таким образом, с ростом длины цепочек из арифметических команд между командами обращения к памяти преимущество в производительности ВПП по сравнению с процессорами традиционной архитектуры только увеличивается.

Заключение

Таким образом, результаты моделирования ВПП с использованием вычислительных ресурсов МСЦ РАН показали, что на программах обработки сеток, таких как программы Stencil, в этом процессоре действительно можно достичь значительно более высокой производительности за счет в 2–3 раза меньшего числа выполняемых команд и использования локальной памяти с ёмкостью на 2 порядка больше по сравнению с ёмкостью регистрового файла.

Список литературы

- [1] *Stencil code*, From Wikipedia, the free encyclopedia. [↑]₄₀₀
- [2] J. Peraza, A. Tiwari, M. Laurenzano, L. Carrington, W. A. Ward, R. Campbell. “Understanding the performance of stencil computations on Intel’s Xeon Phi”, 2013 IEEE International Conference on Cluster Computing (CLUSTER) (23–27 Sept. 2013, Indianapolis, IN, USA). [↑]_{401, 409, 410}
- [3] B. Wah (ed.). “Dataflow Computers: Their History and Future”, *Wiley Encyclopedia of Computer Science and Engineering*, John Wiley & Sons, Inc., 2008. [↑]₄₀₁
- [4] 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. [↑]_{401, 403, 404}
- [5] 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 СКТ-2016, в 2 т. Т. 1 (19–24 сентября 2016 г., Дивноморское, Россия), Изд-во ЮФУ, Ростов-на-Дону, 2016, с. 36–41. [↑]₄₀₂
- [6] Н. И. Дикарев, Б. М. Шабанов, А. С. Шмелёв. «Моделирование параллельной работы ядер векторного потокового процессора с общей памятью», *Программные системы: теория и приложения*, 9:1(36) (2018), с. 37–52. [↑]₄₀₉

Поступила в редакцию 12.11.2018

Переработана 05.12.2018

Опубликована 30.12.2018

Рекомендовал к публикации

Программный комитет

Седьмого национального суперкомпьютерного форума *НСКФ-2018*

Пример ссылки на эту публикацию:

Н. И. Дикарев, Б. М. Шабанов, А. С. Шмелёв. «Недостаточная пропускная способность памяти на программе Stencil: преимущество векторного потокового процессора». *Программные системы: теория и приложения*, 2018, **9:4**(39), с. 399–415.

10.25209/2079-3316-2018-9-4-399-415

http://psta.psiras.ru/read/psta2018_4_399-415.pdf

Об авторах:

Николай Иванович Дикарев

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

0000-0002-7857-3250

e-mail: nic@jscs.ru

Борис Михайлович Шабанов

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

e-mail: shabanov@jscs.ru

Александр Сергеевич Шмелёв

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

0000-0002-1941-7792

e-mail: guest8993@rambler.ru

CSCSTI 50.33.04, 50.33.39

UDC 004.27

Nikolay Dikarev, Boris Shabanov, Aleksandr Shmelev. *Insufficient memory bandwidth on Stencil code: the advantage of a vector dataflow processor.*

ABSTRACT. The main factor limiting performance for the most part of high-performance computing applications is insufficient memory bandwidth, not computational power. Software methods for overcoming this drawback are block methods localizing memory accesses within fast on-chip memory, and “software pipelining” for organizing calculations in the form of chains of arithmetic commands between memory accesses.

Software pipelining was applied to the 2D and 3D Stencil programs in vector dataflow processor. Achieved performance was significantly higher than it is possible to get for the best traditional processors. (*In Russian*).

Key words and phrases: vector processor, dataflow architecture, shared-memory multiprocessor, performance evaluation.

2010 *Mathematics Subject Classification:* 68Q85;65Y05,68W10

References

- [1] *Stencil code*, From Wikipedia, the free encyclopedia. [↑]₄₀₀
- [2] J. Peraza, A. Tiwari, M. Laurenzano, L. Carrington, W. A. Ward, R. Campbell. “Understanding the performance of stencil computations on Intel’s Xeon Phi”, 2013 IEEE International Conference on Cluster Computing (CLUSTER) (23–27 Sept. 2013, Indianapolis, IN, USA). [↑]_{401,409,410}
- [3] B. Wah (ed.). “Dataflow Computers: Their History and Future”, *Wiley Encyclopedia of Computer Science and Engineering*, John Wiley & Sons, Inc., 2008. [↑]₄₀₁
- [4] N. I. Dikarev, B. M. Shabanov, A. S. Shmelev. “Vector Dataflow Processor: performance evaluation”, *Izvestiya YuFU. Tekhnicheskiye nauki*, 2014, no.12(161), Tematicheskiy vypusk: Superkomp’yuternyye tekhnologii, pp. 36–46 (in Russian). [↑]_{401,403,404}
- [5] N. I. Dikarev, B. M. Shabanov, A. S. Shmelev. “Choosing optimal performance of vector dataflow processor core”, *Superkomp’yuternyye tekhnologii*, Materialy 4-y Vserossiyskoy nauchno-tekhnicheskoy konferentsii SKT-2016, v 2 t.. V. 1 (19–24 sentyabrya 2016 g., Divnomorskoye, Rossiya), Izd-vo YuFU, Rostov-na-Donu, 2016, pp. 36–41 (in Russian). [↑]₄₀₂
- [6] N. I. Dikarev, B. M. Shabanov, A. S. Shmelev. “Simulation of multicore vector dataflow processor with shared memory”, *Program Systems: Theory and Applications*, **9**:1(36) (2018), pp. 37–52 (in Russian). [↑]₄₀₉

Sample citation of this publication:

Nikolay Dikarev, Boris Shabanov, Aleksandr Shmelev. “Insufficient memory bandwidth on Stencil code: the advantage of a vector dataflow processor”. *Program Systems: Theory and Applications*, 2018, **9**:4(39), pp. 399–415. (In Russian).

10.25209/2079-3316-2018-9-4-399-415

http://psta.psiras.ru/read/psta2018_4_399-415.pdf