Н.И. Дикарев, Б.М. Шабанов

ВЕКТОРНЫЙ ПОТОКОВЫЙ ПРОЦЕССОР

В последние десятилетия происходил быстрый рост производительности высокопроизводительных вычислительных систем (суперЭВМ) и такое же быстрое расширение сферы их применения. В результате суперЭВМ превратились в мощный инструмент, использование которого позволяет выйти на качественно новый уровень во многих отраслях науки и техники.

Однако, если проанализировать список TOP500 - систем наивысшей производительности, установленных в мире [1], легко заметить, что за последние 3 – 5 лет повышение производительности скалярных многопроцессорных суперЭВМ, которые составляют более 95% списка, происходит в основном за счет совершенствования технологии изготовления используемых в них микропроцессоров и, как следствие, роста их тактовой частоты. Что же касается архитектуры, то производительность скалярных микропроцессоров уже достигла своего предела, который составляет 5 - 6 команд в такт.

Следует отметить, что невозможность дальнейшего увеличения пропускной способности конвейера команд у современных микропроцессоров носит фундаментальный характер и определяется последовательной семантикой программ, свойственной фон-неймановской архитектуре процессора. А именно, команды в программе часто должны выполняться строго последовательно, когда, не вычислив результат текущей команды, нельзя начать выполнение команды следующей, например, если результат предыдущей команды используется в качестве операнда следующей команды или выполняется команда перехода, которая и определяет адрес следующей команды.

В современных суперскалярных микропроцессорах поиск параллелизма среди выполняемых команд осуществляется аппаратными методами, а в микропроцессорах Intel Itanium с длинным командным словом (VLIW) - компилятором. Однако и в том, и в другом типе микропроцессоров при увеличении числа команд, выдаваемых в такт ($K > 4$), возникают труднопреодолимые препятствия. Во-первых, среди этих последовательных K команд, выбираемых из КЭШ команд каждый такт, как правило, встречается команда перехода, и следующие за ней команды чаще всего не выполняются. Во-вторых, аппаратные затраты в ряде устройств процессора, например в схеме регистрового файла, с ростом K увеличиваются пропорционально K^2, что требует соответствующего увеличения времени такта. В результате в микропроцессорах семейства Itanium пропускную способность конвейера команд удалось поднять до 6 команд в такт, и лишь один суперскалярный микропроцессор – IBM Power4, выпущенный в 2002 г., выдает на исполнение до 5 команд в такт. Причем эта же пропускная способность осталась и в микропроцессоре IBM Power5, выпущенном в 2005 г., что лишь подтверждает труднопреодолимый характер встретившихся препятствий.

Кроме того, при выполнении реальных программ производительность микропроцессора часто оказывается существенно ниже его пиковой производительности. Снижение эффективности обусловлено недостаточным быстродействием подсистемы памяти в случае промаха при обращении к КЭШ памяти микропроцессора. Реальная производительность современных микропроцессоров начинает падать, если время выборки из памяти (оперативной) превышает 15 – 20 тактов, а пропускная способность становится меньше 30 – 40 байт в такт. В случае же отсутствия данных в КЭШ памяти микропроцессора время выборки требуемого слова возрастает до ста тактов, а пропускная способность тракта передачи данных снижается до 2 – 4 байт в такт. Причем с ростом тактовой частоты разрыв в быст-

родействии процессора и памяти лишь увеличивается [2]. Отсюда легко понять, что частые промахи при обращении к КЭШ снижают эффективность работы микропроцессора до единиц процентов, и случается это достаточно часто. Так, согласно [3], эффективность работы микропроцессоров IBM Power4, например, составляет лишь 2% - 13% при выполнении 10 из 11 типовых научных задач, и лишь на одной задаче достигает 44%.

Таким образом, одним из основных недостатков скалярных многопроцессорных суперЭВМ является то, что они слишком мало дают пользователю от их пиковой производительности. Гораздо лучше в этом отношении зарекомендовали себя векторные суперЭВМ, которые имеют два важных преимущества. Во-первых, пиковая производительность у векторных процессоров, как правило, в 2 – 4 раза выше по сравнению с современными скалярными микропроцессорами, и, во-вторых, реальная производительность на большинстве научных задач у векторного процессора значительно ближе к его пиковой производительности. Поэтому одной из актуальных задач совершенствования суперЭВМ, особенно для программ, в которых можно задействовать лишь ограниченное число процессоров, является разработка более мощных векторных процессоров.

В Межведомственном суперкомпьютерном центре (МСЦ) РАН ведется разработка векторного процессора с архитектурой управления потоком данных (data-flow), реальная производительность которого может быть повышена в 5 – 8 раз по сравнению с современными векторными процессорами и почти на два порядка величины – по сравнению с микропроцессорами.

Разрабатываемый векторный потоковый процессор (ВПП) имеет наиболее совершенную – динамическую модель потоковой архитектуры, которая позволяет выдавать команды на исполнение по мере готовности их операндов с возможностью одновременного выполнения различных итераций вложенных циклов и различных запусков процедур на одном и том же графе программы. Поиск готовых к выполнению команд в динамических потоковых ЭВМ производится в ассоциативной памяти (АП) по совпадению признаков у токенов операндов, поскольку парные операнды должны иметь не только один и тот же номер выполняемой команды, но и одинаковые поля индекса, номера итерации и запуска процедуры в признаке (контексте) токена. В таких ЭВМ переполнение АП недопустимо, поэтому АП должна иметь большую емкость и в то же время быть быстродействующей, что трудно реализовать на практике.

Введение векторной обработки в динамическую потоковую ЭВМ позволяет существенно снизить требования к емкости АП, поскольку одна векторная команда выполняет не одну, а группу из VL элементарных операций, где VL – длина вектора. При хранении векторов в памяти векторов (ПВ) для выполнения векторной команды требуется поиск в АП лишь одной пары токенов с указателями векторов вместо VL пар токенов при скалярной обработке, и требования к емкости АП на векторизуемой части программы снижаются в VL раз.

В отличие от известных проектов векторных потоковых ЭВМ, в которых арифметические операции над векторами выполняются лишь для векторов в векторных регистрах, в предлагаемом ВПП блок векторных регистров отсутствует и используется одноуровневая ПВ большой емкости (уровня оперативной памяти). При этом исключается проблема возможности переполнения регистрового файла, которая не имеет простого решения, если учесть полную асинхронность потоковой модели вычислений. Так же не просто в процессе выполнения программы обойтись без участия операционной системы при выделении места под создаваемые массивы в основной памяти. Поэтому распределение ресурса ПВ в ВПП предлагается реализовать на аппаратном уровне.

В качестве единицы фрагментации при аппаратном распределении памяти в ВПП используется вектор с фиксированным числом слов VL=2^n (VL=256 – 1024). Причем большие массивы предлагается хранить в ПВ в виде «векторов-указателей», то есть векторов, элементами которых являются указатели векторов подмассивов, как это показано на рис. 1.

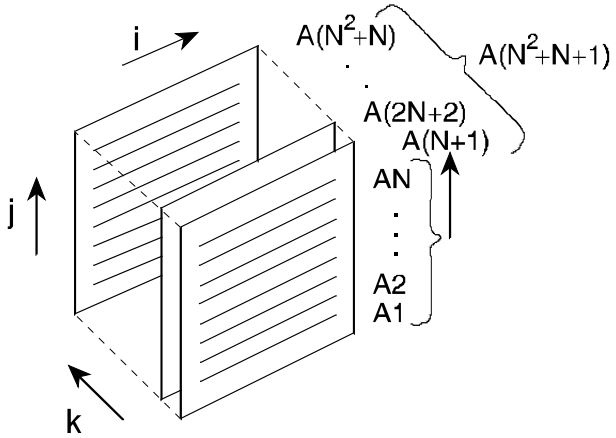

Рис. 1. Представление трехмерного массива векторами-указателями

Тогда аппаратное распределение ресурса ПВ легко осуществить за счет ведения списка свободных векторов. Новый вектор, необходимый для записи результата векторной команды, удаляется из списка свободных векторов, а после выполнения всех команд, в которых этот вектор используется в качестве операнда, он возвращается обратно в список свободных векторов. В результате размещение векторов и массивов в ПВ происходит гибко по мере их создания (уничтожения) и без участия операционной системы.

Такое хранение массивов в одноуровневой ПВ позволяет исключить из графа программы для ВПП команды обращения к памяти (команды пересылки данных между основной оперативной памятью и блоком векторных регистров) и команды, осуществляющие адресные вычисления. В результате общее число выполняемых команд в программах сокращается примерно вдвое, а доля операций с плавающей запятой в них может быть доведена до 70 - 99 %. Кроме того, представление массивов в виде векторов-указателей дает возможность выполнять одинаковые операции над всеми их элементами, например над строками, содержащимися в векторе-указателе матрицы. Тем самым удается векторизовать не один, а два вложенных цикла программы. Рассмотрим в качестве примера фрагмент программы обработки матриц, в котором элементы матрицы результата B вычисляются из элементов матрицы A по следующему алгоритму:

```
DO 20 i=1, N
    DO 20 j=1, N
    20  B(i,j)=A(i,j) - A(k,j)*A(i,k).
```

В тесте LINPACK (решение системы линейных уравнений методом исключения Гаусса) основной объем вычислений приходится на аналогичный алгоритм обработки матриц. Внутренний цикл в этой программе исключается за счет использования векторных арифметических команд, и программу можно представить в следующем виде:

```
DO 20 i=1, N
Bi = Ai - Ak * A(i,k).
```

Граф этой программы для ВПП приведен на рис. 2. В нем для вычисления векторов строк Bi матрицы B используются одинаковые операции над строками матрицы A и все необходимые токены векторов строк Ai, участвующие в цикле, с именами A1,A2,...AN создаются одной командой "Формирование Потока" (ФП).

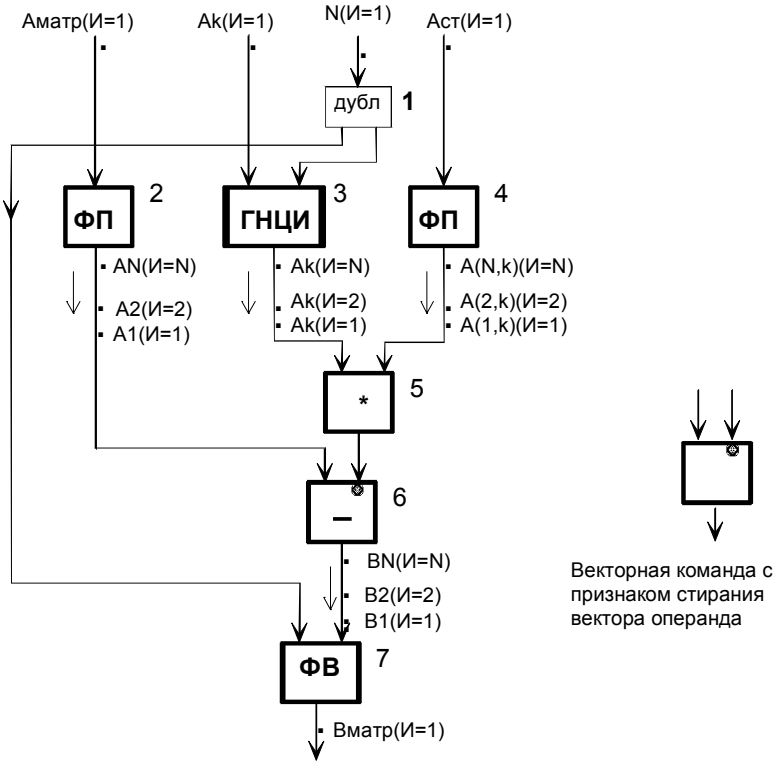

Рис. 2. Граф программы для потоковой ЭВМ с векторной обработкой

Эта команда читает из ПВ элементы вектора указателя Аматр (A(N+1) на рис. 1) и формирует последовательность токенов A1,A2,...AN. Причем в поле И признака этих токенов команда ФП записывает индексы от 1 до N, что позволяет выполнять итерации цикла параллельно за счет поиска в АП готовых пар операндов для команд в теле цикла по совпадению индекса И (помимо номера команды К). Аналогично другая векторная команда ФП формирует для всех итераций цикла по i от 1 до N элементы A(i,k) из вектора Аст (команда 4 на рис. 2). Здесь считается, что вектор Аст, состоящий из элементов k-го столбца матрицы A, предварительно сформирован командой "Сборка вектора". Для размножения N раз указателя вектора Ak используется команда 3 ГНЦИ - "Групповое Начало Цикла по Итерации". Эта команда порождает сразу для всех итераций копии "константы цикла", записывая в поле И признака значения от 1 до N. Наконец, команда 7 "Формирование Вектора" (ФВ) в графе на рис. 2 по приходе токена на левый вход резервирует буферный векторный регистр на N слов и выделяет место в ПВ для записи его содержимого. Затем приходящие на правый вход команды 7 токены с указателями векторов строк Bi записываются в элементы буферного регистра в соответствии с их индексами И. После записи последнего элемента в буферный векторный регистр производится перезапись его содержимого в ПВ и выдача токена результата команды, т.е. вектора указателя матрицы Вматр.

Выдача токена результата команды ФВ свидетельствует о выполнении всех итераций цикла, причем в приведенном на рис. 2 графе программы аппаратно реализована не только барьерная синхронизация для выхода из внешнего цикла, но и изменение индексов для выполняемых в теле цикла итераций. Именно это и позволяет утверждать, что в приведенном графе программы векторизованы оба вложенных цикла исходной программы.

Легко заметить, что в графе на рис. 2 векторные команды 5 и 6 для обработки чисел с плавающей запятой выполняются по N раз каждая. Причем для организации выполнения этих команд требуется лишь однократная выдача команд ФП, ГНЦИ и ФВ, векторизующих внешний цикл. В результате, на $2N^2$ операций с плавающей запятой в данном графе программы приходится лишь одно выполнение скалярной команды дублирования 1, поскольку все остальные команды, включая ФП, ГНЦИ и ФВ, являются групповыми (векторными). Соответственно, степень векторизации данной программы практически равна 1, а доля операций с плавающей запятой достигает 99% от общего числа выполняемых операций уже при длине вектора N=256.

Векторизация двух вложенных циклов в программах приводит к тому, что производительность скалярной обработки и управления в ВПП уже не ограничивает производительность его векторной обработки и она может вестись страницами по 64 – 128 слов в такт, в то время как в традиционных векторных процессорах она составляет 8 – 16 слов в такт.

К настоящему времени в МСЦ РАН разработаны схемы основных узлов ВПП, на основе которых написана VHDL-модель процессора уровня регистровых передач. Такая модель позволяет оценить время выполнения программ в ВПП с точностью до такта. На рис.2 приведены зависимости реальной производительности ВПП на тесте LINPACK от размера матрицы *N* и пиковой производительности ВПП, полученные в результате экспериментального исследования модели ВПП [4].

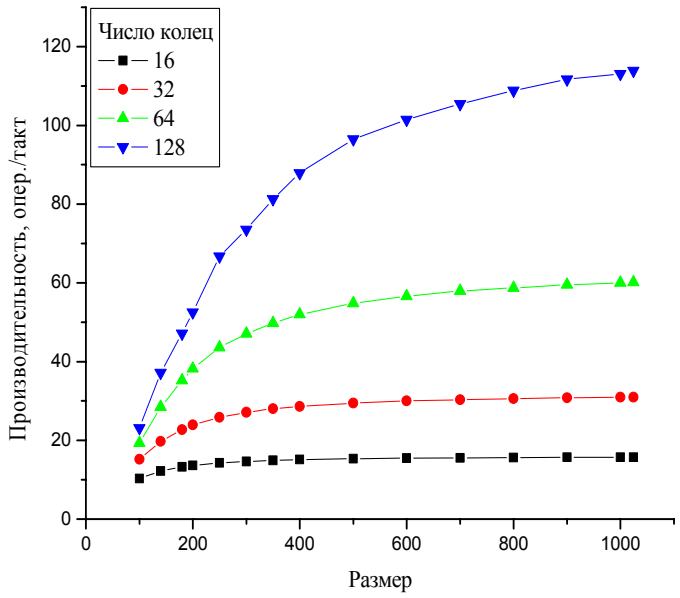

Рис.2. Зависимость производительности ВПП от размера матрицы исходных данных

Пиковая производительность ВПП задается в качестве параметра модели числом “колец” обработки *M* в векторном блоке процессора, причем каждое ”кольцо” содержит модуль ПВ и конвейерное АЛУ, что позволяет вести обработку векторов страницами по *M* элементов в такт. Соответственно, пиковая производительность ВПП равна *M* операций с плавающей запятой (флоп) в такт, а реальная производительность легко вычисляется по времени выполнения теста в тактах, поскольку общее число операций с плавающей запятой в тесте равно $2/3*N^3+2N^2$.

Как видно из зависимостей на рис. 2, реальная производительность ВПП приближается к пиковой с увеличением размера матрицы *N*, причем, чем выше пиковая производительность, тем больше и размер матрицы, на котором достигается заданная эффективность. Такой характер зависимостей справедлив и для известных процессоров. Отличие в том, что ВПП обеспечивает значительно более высокую реальную производительность (при той же эффективности), чем существующие векторные процессоры. Так, векторный процессор NEC SX-6 с пиковой производительностью 16 флоп в такт на матрице 100*100 имеет эффективность 14,5% [5], и его реальная производительность равна 2,32 флоп в такт. На том же размере матрицы ВПП с пиковой производительностью 128 флоп в такт имеет эффективность 17,5%, т.е. его реальная производительность почти в 10 раз выше.

Таким образом, результаты тестирования модели ВПП с помощью теста LINPACK показывают, что ВПП действительно может обеспечить на порядок более высокую реальную производительность по сравнению с существующими векторными процессорами и на два порядка – по сравнению со скалярными микропроцессорами.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Top 500 ranking of the fastest computers. http://www.top500.org/.
2. *Дикарев Н.И., Шабанов Б.М.* Реальная и пиковая производительности суперЭВМ // Автоматизация проектирования. № 1-2 (13). М., 2000. С.3-15.
3. *Oliker L. et al.* Evaluation of Cache-based Superscalar and Cacheless Vector Architectures for Scientific Computations. Proc. Int. Conf. on Supercomputing, Nov. 2003, 26 pp.
4. *Аблязов Н.А., Дикарев Н.И., Макаров О.С., Шабанов Б.М.* Результаты выполнения теста LINPACK на модели векторного потокового процессора // «Высокопроизводительные вычислительные системы и микропроцессоры». Труды ИМВС РАН. Вып. 6, М., 2004. С.42-55.
5. LINPACK Benchmark. http://www.netlib.org/performance/html/ /linpack.data.col0.html, http://www.netlib.org/benchmark/performance.ps.

С.В. Торчигин

УПРАВЛЕНИЕ ВЫЧИСЛИТЕЛЬНЫМИ ПРОЦЕССАМИ В ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЕ С АВТОМАТИЧЕСКИМ РАСПРЕДЕЛЕНИЕМ РЕСУРСОВ

В настоящее время существует большое количество задач, требующих производительности вычислительных систем порядка $10^{12} – 10^{16}$ операций в секунду [1]. Основным методом увеличения производительности вычислительных систем становится метод распараллеливания вычислительных процессов на математическом уровне. Однако это не решает всей проблемы в целом. В этой связи возникает потребность в разработках вычислительных систем нетрадиционной архитектуры. К таким системам относится вычислительная система с автоматическим распределением ресурсов (ВСАРР) [2]. В основе этой вычислительной системы лежит ме-