УДК 519.6

ИССЛЕДОВАНИЕ БЛОЧНО-ЦИКЛИЧЕСКИХ АЛГОРИТМОВ НА СЕМЕЙСТВЕ КЛАСТЕРОВ СКИТ

А.Н. Химич, А.В. Попов, Т.В. Чистякова, О.В. Рудич, Т.А. Герасимова

Институт кибернетики им. В.М. Глушкова НАН Украины
03680, Киев-187, проспект Академика Глушкова, 40,
e-mail: dept150@insyg.kiev.ua, тел.: 266 1196; факс: (044) 266 7418

Рассмотрены некоторые особенности создания циклических алгоритмов для параллельных компьютеров и тенденции их развития. Проведены исследования блочно-циклических алгоритмов решения задач линейной алгебры на примере библиотеки программ ScaLAPACK, функционирующей на семействе компьютеров СКИТ, разработанном в Институте кибернетики им. В.М. Глушкова НАН Украины.

Characteristic features of the cyclical algorithms' creation for parallel computers as well as trends of their development are dealt with. Block cyclical algorithms for the solving of the linear algebra problems have been investigated on the example of ScaLAPACK program library functioning on СКИТ computer family created at V.M. Glushkov Institute of cybernetics.

Введение

Вычислительная техника – основа общества научно-технического прогресса. Несмотря на то, что мир насыщен разнообразной вычислительной техникой, остаются актуальными проблемы создания и использования высокопроизводительных компьютеров, производительность которых достигает 10^{13} операций в секунду [1].

Технологический вызов, связанный с перспективой освоения кристаллов, содержащих от 100 млн. до 1 млрд. транзисторов, может быть поддержан только за счет новых идей в области архитектуры, системо-техники, новых вычислительных методов, алгоритмических и программных модулей. Так, прогнозируется, что число транзисторов на кристалле за период 1999 – 2012 гг. увеличится в 70 раз, а тактовая частота – лишь в 2,5 раза.

Одним из путей сокращения времени решения научных и инженерных задач является разработка эффективных алгоритмов и создание на их основе интеллектуального программного обеспечения для исследования и решения научно-технических задач.

Под эффективным алгоритмом решения научно-технической задачи будем понимать алгоритм, позволяющий получить достоверное решение задачи с минимальным использованием оперативной памяти за возможно минимальное время.

Так, учет в алгоритмах и вычислительных схемах особенностей архитектуры и структуры MIMD-компьютера Parsytec Power Xplorer/4 лишь на операции произведения матриц позволил сократить время решения задачи в 20 раз (известно, что вычисление произведения двух матриц можно реализовать шестью различными алгоритмами, причем время их реализации на одном и том же компьютере будет различным, а некоторые алгоритмы из-за больших погрешностей вычислений могут давать ошибочный машинный результат при одной и той же длине машинного слова).

Тенденции развития блочно-циклических алгоритмов

Основной целью при создании параллельных алгоритмов является достижение по возможности наибольшего сокращения времени решения задач. Однако на практике максимальное сокращение времени (в количество раз, близкое к количеству процессоров) можно получить только для задач, по существу тривиальных. Главные факторы, препятствующие этому, таковы:

- отсутствие полного параллелизма в алгоритмах исследования и решения задач (некоторые операции алгоритмов являются последовательными);
- неравномерная загрузка процессоров по числу арифметических операций (несбалансированность нагрузки процессоров);
- коммуникационные потери, обусловленные необходимостью обмена информацией между процессорами и синхронизацией вычислительной системы.

Очевидно, что большинство алгоритмов представляет собой комбинацию фрагментов с различными степенями параллелизма: от минимальной, когда работает один процессор, до максимальной, когда одновременно могут выполняться все операции. Таким образом, большинство алгоритмов является попеременно параллельно-последовательными по своей природе.

Для оценки качества параллельных алгоритмов пользуются, как правило, такими критериями как коэффициент ускорения и коэффициент эффективности.

Разработка высокоэффективных параллельных алгоритмов невозможна без выполнения условий согласованности работы процессоров. Суть этих условий заключается в минимизации коммуникационных потерь, т.е. ожиданий, связанных с завершением вычислений, и обменов данными.

Таким образом, при разработке и реализации параллельных алгоритмов к уже рассмотренным проблемам компьютерных вычислений добавились новые, связанные с распараллеливанием вычислений, а именно:

- распараллеливание не только арифметических действий, а и обменов данными между процессорами;
- определение количества процессоров и топологии межпроцессорных связей, необходимых для эффективного решения задачи;
- обеспечение равномерной загрузки всех процессоров, которые используются для решения задачи;
- организация обменов данными между процессорами;
- синхронизация обменов данными между процессорами;
- минимизация обменов данными между процессорами;
- распределение исходной информации между процессорами в соответствии с выбранной конфигурацией и др.

На сегодняшний день можно сделать вывод, что прототипами большинства известных параллельных алгоритмов за редким исключением (например, в случае ленточных матриц с узкой шириной ленты при решении систем линейных алгебраических уравнений) являются известные последовательные алгоритмы. При таком подходе фактически в каждом процессоре одновременно реализуются последовательные вычислительные схемы над локальным блоком данных, полученным в результате декомпозиции исходных данных по тому или иному способу. Принципиально важно, что при таком адаптивном подходе для параллельных алгоритмов сохраняется преемственность результатов, полученных в теории численных методов (точность, сходимость, сложность и др.).

С этой точки зрения характерна эволюция, которую претерпели параллельные алгоритмы (в задачах линейной алгебры), основанные на элементарных ортогональных преобразованиях: от алгоритмов, базирующихся на элементарных односторонних и двусторонних преобразованиях Якоби (как известно, не лучших по экономичности представителях последовательных алгоритмов), к методам, основанным на преобразованиях Гивенса, Хаусхолдера – методам, стоящим во главе иерархии списка по эффективности среди ортогональных последовательных алгоритмов. По-видимому, это обстоятельство связано, с одной стороны, с естественным параллелизмом алгоритмов, основанных на вращениях Якоби, а с другой стороны, с влиянием эффекта Гайдна [2], понижающим эффективность методов Гивенса и Хаусхолдера для задач линейной алгебры при обычном способе хранения матриц (одномерным блочным столбцовым или строчным способом распределения, рис. 1). В этом случае каждый процессор содержит только один блок столбцов матрицы. Столбец k хранится на процессоре k/t, где $t = n/p$ – максимальное число столбцов, хранимых в процессоре (на рис. 1 $n = 16$ и $p = 4$). Эта схема не обеспечивает хорошей сбалансированности загрузки процессоров, потому, что как только первые t столбцов будут обработаны, 0-й процессор закончит работу, после обработки следующих t столбцов закончит работу 1-й процессор и т.д.. Аналогично не дает хорошей сбалансированности загрузки процессоров и одномерное блочное распределение строк по процессорам.

Значительным шагом к улучшению сбалансированности загрузки процессоров, а, следовательно, и эффективности вычислительных алгоритмов стала идея циклического способа хранения и обработки матриц, который приводит к сбалансированной схеме для алгоритмов триангуляции или трехдиагонализации матриц. Идея столбцово (строчно) циклического способа хранения и обработки матриц (рис. 2) была независимо высказана в нескольких работах (см., например, [3], в том числе и в работе авторов [4]). В соответствии, например, со строчно-циклической схемой матрица распределяется по p процессорах следующим образом: в i-ом процессоре располагаются строки с номерами $i, i+p, i+2p, \ldots$. Эта схема, сохраняя тот же, что и для последовательных алгоритмов триангуляции или трехдиагонализации матриц, порядок обработки строк (столбцов), предусматривает изменение на единицу номера процессора при переходе от одной строки к следующей. Таким образом, достигается примерно одинаковый объем вычислений в каждом процессоре при реализации алгоритмов, т.е. практически исключается влияние эффекта Гайдна.

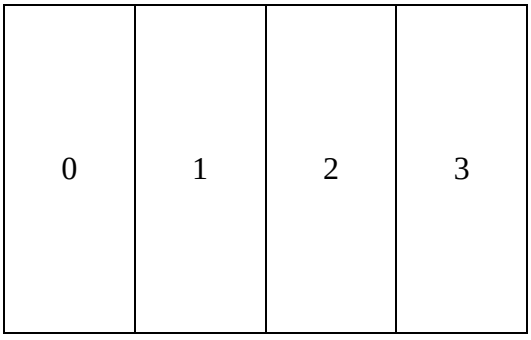

Рис. 1

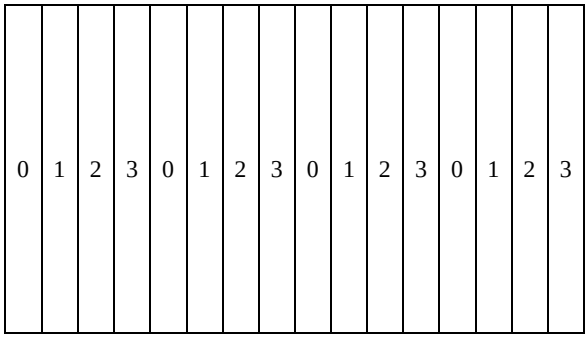

Рис. 2

Такая ситуация характерна не только для алгоритмов, основанных на ортогональных преобразованиях, но и для алгоритмов Гаусса, Холесского и других алгоритмов приведения матрицы к одному из стандартных видов, для которых характерен постепенно уменьшающийся от шага к шагу размер обрабатываемой матрицы.

Показанная на рис. 3 третья схема – столбцовая блочно-циклическая**,** является компромиссом между дистрибутивными схемами, показанными на рис. 1, 2. Выбирая размер блока n_b, делим столбцы по группам размера n_b, и распределяем эти группы циклическим способом (на рис. 3 $n = 16$, $p = 4$ и $n_b = 2$**)**. Это означает, что столбец k хранится в процессоре с номером $[(k\text{-}1)/\ n_b]\text{mod}\ p$. Фактически, эта схема включает первые две схемы хранения как специальные случаи для $n_b = t = n/p$ и $n_b = 1$. Для $n_b > 1$ такая схема дает несколько худшую сбалансированность загрузки процессоров, по сравнению с циклическим распределением столбцов, но зато уменьшается общее время латентности системы, поскольку уменьшается количество соединений между процессорами для обменов данными.

Четвертая схема (рис. 4), двумерное блочно-циклическое распределение столбцов и строк, включает все предыдущие схемы как специальные случаи. Одним из важных факторов целесообразности такого распределения и обработки матриц, наряду с хорошими свойствами сбалансированности и латентности, является возможность использования процедур стандартных библиотечных программ BLAS, в частности библиотеки BLAS 3 из [5].

0	1	2	3	0	1	2	3

Рис. 3

0	1	0	1	0	1	0	1
2	3	2	3	2	3	2	3
0	1	0	1	0	1	0	1
2	3	2	3	2	3	2	3
0	1	0	1	0	1	0	1
2	3	2	3	2	3	2	3
0	1	0	1	0	1	0	1
2	3	2	3	2	3	2	3

Рис. 4

Функциональные возможности библиотеки программ Scalapack

Основными сдерживающими факторами широкого использования MIMD-компьютеров являются как отсутствие развитого прикладного программного обеспечения, так и трудности, связанные с использованием параллельных компьютеров пользователями, имеющими опыт работы только на компьютерах традиционной архитектуры.

Преодолению этих трудностей служит идея создания портабельного программного обеспечения. Для решения систем линейных алгебраических уравнений на однопроцессорных компьютерах эта идея основана на создании библиотеки процедур, реализующих базисные операции линейной алгебры, BLAS (basic linear algebra subroutine) и реализована в пакете программ LINPACK. Дальнейшее развитие идеи создания портабельного программного обеспечения для архитектур параллельных компьютеров привело к созданию пакета программ LAPACK на основе библиотек BLAS2 и BLAS3. Для компьютеров MIMD-архитектуры BLAS3 основана на блочных аналогах последовательных алгоритмов решения задач линейной алгебры. Пакет программ ScaLAPACK [5] по линейной алгебре для компьютеров с распределенной памятью (distributed memory) реализован с использованием BLAS версий 1, 2, 3 совместно с библиотекой программ BLACS (basic linear algebra communication subroutine) и обменом информацией посредством передачи сообщений, поддерживающих PVM (Parallel Virtual Machine) или MPI (Massage Passing Interface). Это – продолжение проекта LAPACK [6] для параллельных компьютеров с разделяемой памятью (shared memory).

На рис. 5 показана общая структура ScaLAPACK, на котором компоненты библиотеки, выше расположенные разделительной линии, содержат подпрограммы, которые выполняются параллельно на некотором наборе процессоров и в качестве аргументов используют векторы и матрицы распределенные по этим процессорам. Компоненты, которые расположенные ниже разделительной линии, вызываются на одном процессоре и работают с локальными данными. Каждый из компонентов ScaLAPACK – это независимая библиотека подпрограмм, которая не является частью библиотеки, но необходима для ее работы.

Функциональные подпрограммы ScaLAPACK, реализующие алгоритмы решения задач, созданы на основе библиотеки LAPACK как наиболее полно отвечающей архитектуре современных процессоров (программы оптимизированы для эффективного использования кэш-памяти).

В библиотеке BLAS содержатся подпрограммы первого, второго и третьего уровней, реализующие базовые операции линейной алгебры (такие как: действия над векторами и скалярами, умножение матрицы на вектор или произведение двух матриц), которые могут использоваться на векторных суперкомпьютерах и параллельных компьютерах с разделяемой памятью. Подпрограммы пакета PBLAS содержат аналогичные подпрограммы трех уровней с параллельной организацией вычислений при использовании подпрограмм библиотек BLAS и BLACS**.** BLACS является библиотекой подпрограмм, предназначенных для работы с параллельными процессами.

ScaLAPACK функционирует в операционной системе Linux и поддерживается системой параллельных вычислений MPI.

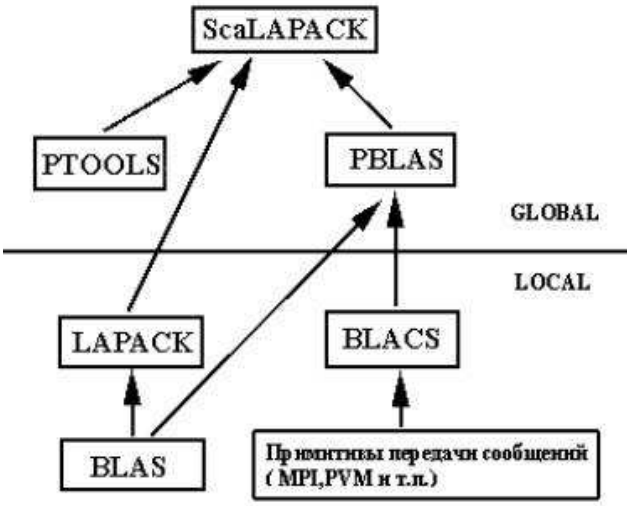

Рис. 5

Все подпрограммы ScaLAPACK реализованы на языке Fortran77 в арифметике вещественных чисел одинарной или двойной точности и в комплексной арифметике, также одинарной или двойной точности.

В состав библиотеки включены 530 подпрограмм, которые разделяются на три категории:

- драйверные подпрограммы, каждая из которых решает некоторую задачу, например, решение системы линейных алгебраических уравнений или нахождение собственных значений вещественной симметричной матрицы (таких подпрограмм 14 для каждого типа данных);
- вычислительные подпрограммы, реализующие логически законченные части алгоритма, например, LU–разложение матрицы или приведение вещественной симметричной матрицы к трехдиагональному виду;
- служебные подпрограммы, которые выполняют некоторые внутренние вспомогательные действия.

ScaLAPACK поддерживает работу со следующими видами матриц: плотными прямоугольными или квадратными, ленточными, трехдиагональными. Для каждого из этих видов строится сетка процессоров различной топологии и используется различная схема распределения данных по процессорам.

Для плотных матриц используется двумерная сетка процессоров. На рис. 6 показан пример, такой, двумерной сетки размерностью 2×3 из 6 процессоров.

	0	1	2
0	0	1	2
1	3	4	5

Рис. 6

Для плотных матриц принят блочно-циклический способ распределения данных по процессорам. При таком распределении матрица разбивается на блоки размера *MB* x *NB*, где *MB* – количество строк в блоке, а *NB* – столбцов, и эти блоки циклически распределяются по процессорам.

Это показано на конкретном примере распределения матрицы общего вида *A*(9,9), т.е. $M = 9$, $N = 9$, на сетке процессоров $NPROW \times NPCOL = 2 \times 3$ при условии, что размер блока $MB \times NB = 2 \times 2$. Разбиение исходной матрицы на блоки 2×2 показано на рис. 7.

a_{11} a_{12} a_{21} a_{22}	a_{13} a_{14} a_{23} a_{24}	a_{15} a_{16} a_{25} a_{26}	a_{17} a_{18} a_{27} a_{28}	a_{19} a_{29}
a_{31} a_{32} a_{41} a_{42}	a_{33} a_{34} a_{43} a_{44}	a_{35} a_{36} a_{45} a_{46}	a_{37} a_{38} a_{47} a_{48}	a_{39} a_{49}
a_{51} a_{52} a_{61} a_{62}	a_{53} a_{54} a_{63} a_{64}	a_{55} a_{56} a_{65} a_{66}	a_{57} a_{58} a_{67} a_{68}	a_{59} a_{69}
a_{71} a_{72} a_{81} a_{82}	a_{73} a_{74} a_{83} a_{84}	a_{75} a_{76} a_{85} a_{86}	a_{77} a_{78} a_{87} a_{88}	a_{79} a_{89}
a_{91} a_{92}	a_{93} a_{94}	a_{95} a_{96}	a_{97} a_{98}	a_{99}

Рис. 7

После циклического распределения блоков вдоль строк и столбцов сетки процессоров 2×3 получаем следующее распределение матричных элементов по процессорам (рис. 8).

	0				1			2	
0	a_{11}	a_{12}	a_{17}	a_{18}	a_{13}	a_{14}	a_{19}	a_{15}	a_{16}
	a_{21}	a_{22}	a_{27}	a_{28}	a_{23}	a_{24}	a_{29}	a_{25}	a_{26}
	a_{51}	a_{52}	a_{57}	a_{58}	a_{53}	a_{54}	a_{59}	a_{55}	a_{56}
	a_{61}	a_{62}	a_{67}	a_{68}	a_{63}	a_{64}	a_{69}	a_{65}	a_{66}
	a_{91}	a_{92}	a_{97}	a_{98}	a_{93}	a_{94}	a_{99}	a_{95}	a_{96}
1	a_{31}	a_{32}	a_{37}	a_{38}	a_{33}	a_{34}	a_{39}	a_{35}	a_{36}
	a_{41}	a_{42}	a_{47}	a_{48}	a_{43}	a_{44}	a_{49}	a_{45}	a_{46}
	a_{71}	a_{72}	a_{77}	a_{78}	a_{73}	a_{74}	a_{79}	a_{75}	a_{76}
	a_{81}	a_{82}	a_{87}	a_{88}	a_{83}	a_{84}	a_{89}	a_{85}	a_{86}

Рис. 8

Для ленточных и трехдиагональных матриц строится одномерная сетка процессоров $1 \times NPROCS$, где $NPROCS$ – количество используемых процессоров. В этом случае используется блочный принцип распределения. В каждом процессоре хранится только один блок, размер которого выбирается из соображений равномерности распределения, т.е. $NB \approx N / NPROCS$. При этом объекты левой части уравнения распределяются по столбцам, а правой – по строкам, т.е. для правых частей уравнений с ленточными и трехдиагональными матрицами используется транспонированная одномерная сетка $NPROCS \times 1$. На рис. 9 показан пример для ленточной матрицы $A(7,7)$.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & a_{24} & 0 & 0 & 0 \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & 0 & 0 \\ 0 & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} & 0 \\ 0 & 0 & a_{53} & a_{54} & a_{55} & a_{56} & a_{57} \\ 0 & 0 & 0 & a_{64} & a_{65} & a_{66} & a_{67} \\ 0 & 0 & 0 & 0 & a_{75} & a_{76} & a_{77} \end{pmatrix}$$

Рис. 9

Схема хранения матричных элементов несимметричной ленточной матрицы в 3-х процессорах при условии $NB = 3$ показана на рис. 10. Звездочками отмечены неиспользуемые позиции.

0			1			2
*	*	a_{13}	a_{24}	a_{35}	a_{46}	a_{57}
*	a_{12}	a_{23}	a_{34}	a_{45}	a_{56}	a_{67}
a_{11}	a_{22}	a_{33}	a_{44}	a_{55}	a_{66}	a_{77}
a_{21}	a_{32}	a_{43}	a_{54}	a_{65}	a_{76}	*
a_{31}	a_{42}	a_{53}	a_{64}	a_{75}	*	*

Рис. 10

Исследование эффективности блочных алгоритмов на семействе СКИТ

Исследования блочных алгоритмов, реализованных в ScaLAPACK, показали, что они наилучшим образом соответствуют топологии межпроцессорных связей «решетка», что дает возможность эффективно реализовывать обмены информацией между процессорами. Однако для каждого метода решения и для каждого порядка матрицы существует свое оптимальное количество процессоров. Диаграмма, которая показана на рис. 11, иллюстрирует время решения системы линейных алгебраических уравнений с симметричной

положительно определенной матрицей методом Холесского на различном количестве процессоров. В этом случае оптимальное число процессоров – 8.

Кроме того, блочные алгоритмы позволяют варьировать размеры блоков, на которые делятся исходные матрицы. Таким образом, размеры локальных подматриц можно подобрать так, чтобы блок матрицы, который многократно используется в матрично-векторных операциях, помещался в кэш-память. Это дает возможность значительно сократить количество обращений к основной оперативной памяти. Диаграмма на рис. 12 демонстрирует как изменяется время решения системы линейных алгебраических уравнений с плотной невырожденной матрицей методом Гаусса на одном процессоре при различном количестве строк в блоке.

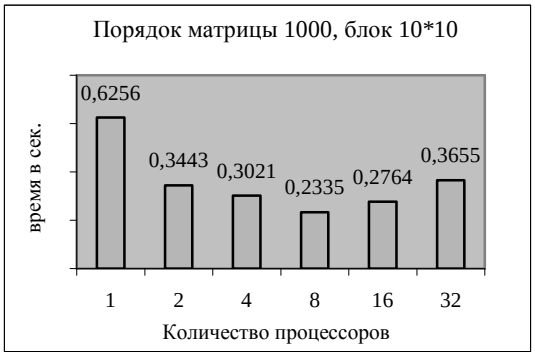

Рис. 11

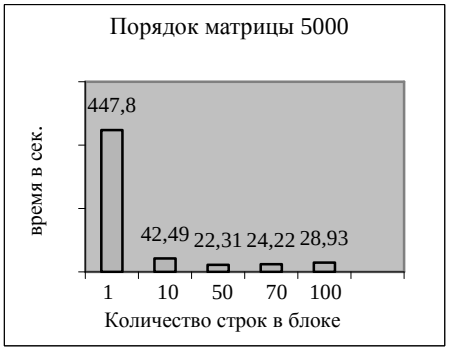

Рис. 12

Диаграммы, показанные на рис. 13, 14, демонстрируют зависимость времени решения системы линейных алгебраических уравнений с вырожденной матрицей методом наименьших квадратов от количества процессоров и размеров блоков. Здесь приведены результаты исследований на СКИТ–2, коммутационная сеть SCI, компилятор Intel, версия MPI – Scali.

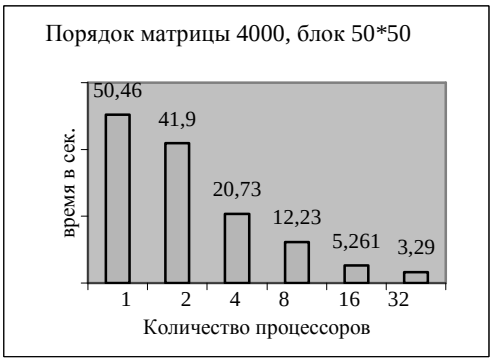

Рис. 13

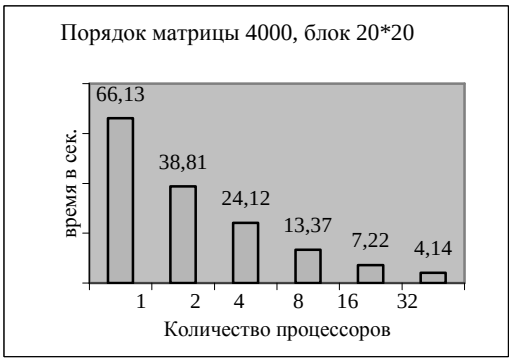

Рис. 14

Значительный прирост производительности алгоритмов дает также эффективное использование конвейеризации инструкций. Известно, что все современные процессоры имеют конвейер инструкций. Конвейерный процессор принимает новую инструкцию каждый цикл, даже если предыдущие инструкции не завершены. Выполнение нескольких инструкций перекрывается – в процессоре, находятся несколько инструкций на разных стадиях готовности (выбор, декодирование, исполнение, запись результатов). Конвейер инструкций может уменьшить количество циклов на инструкцию, если будет одновременно выполнять несколько инструкций, которые находятся на разных стадиях. Для правильного заполнения конвейера в программах целесообразно увеличивать длину линейных участков (без команд перехода) так, чтобы она была больше глубины конвейера. С этой целью имеет смысл операторы цикла в матрично-векторных операциях писать с “разворачиванием”. В ScaLAPACK в подпрограммах библиотек BLAS и PBLAS, которые реализуют матрично-векторные операции, разворачивание циклов реализовано так, чтобы за один проход цикла вычислялось сразу 4 или 5 элементов.

Заключение

Проведенные на семействе кластеров СКИТ исследования блочно-циклических алгоритмов для решения задач линейной алгебры, реализованных в ScaLAPACK, выявили высокую эффективность таких алгоритмов. При этом определяющими являются следующие факторы:

- соответствие алгоритма коммуникационной среде и топологии межпроцессорных связей;
- оптимальные размеры блоков матриц, согласованные с объемом кэш-памяти;

– оптимальное количество процессоров для каждой задачи и для каждого размера блоков, на которые разбивается матрица;

– использование оптимизирующих стилей программирования с учетом конвейера инструкций, архитектуры процессоров и возможностей компилятора для тех участков алгоритмов, которые требуют наибольшего времени выполнения.

Однако большинство этих факторов должен учитывать сам пользователь ScaLAPACK. Кроме того, на пользователя возлагается распределение данных по процессорам, которое, как отмечалось выше, не является тривиальным. Эти трудности могут быть преодолены с помощью автоматизации процесса исследования и решения задачи, когда по выявленным компьютером математическим свойствам решаемой задачи автоматически выбирается экономичный алгоритм решения, а под него формируется (опять-таки автоматически) необходимая топология межпроцессорных связей MIMD-компьютера. Такая последовательность действий реализуется в интеллектуальном программном обеспечении для исследования и решения задач [7], [8], в том числе, и для параллельных компьютеров с реализацией принципа скрытого параллелизма.

Интеллектуальное программное обеспечение, реализующее на MIMD-компьютере принцип скрытого параллелизма, позволяет освободить конечного пользователя от проблем, связанных с организацией параллельных вычислений и обеспечивает автоматизацию следующих процессов: распараллеливание алгоритмов; выбор количества процессоров и других параметров алгоритма (например, размеры блоков матриц), обеспечивающих эффективное решение задачи; создание конфигурации вычислительной системы из процессоров; распределение исходной информации по процессорам; реализацию обменов информацией между процессорами; синхронизацию вычислительной системы.

1. *Hamilton S.* Taking Moore's law into the new century // Computer – 1999. – N 1. – P. 12–16.
2. *Валях Е.* Последовательно-параллельные вычисления. – М.: Мир, 1985. – 456 с.
3. *Ортега Дж.* Введение в параллельные и векторные методы решения линейных систем. – М.: Мир, 1991. – 368 с.
4. *Михалевич В.С., Бик Н.А., Брусникин Б.Н.,…, Химич А.Н. и др. / Под ред. И.Н. Молчанова.*Численные методы для многопроцессорного вычислительного комплекса ЕС. – М.: Издание ВВИА им. Н.Е. Жуковского, 1986. – 401 с.
5. http://www.netlib.org/scalapack
6. http://www.netlib.org/lapack
7. *Молчанов И.Н.* Проблемы и перспективы развития прикладного программного обеспечения // Управляющие системы и машины. – 1988. – № 1. – С. 56 –61.
8. *Молчанов І.М., Галба Є.Ф., Попов О.В., Хіміч О.М., Чистякова Т.В., Яковлев М.Ф.* Інтелектуальний інтерфейс для дослідження та розв'язування задач обчислювальної математики з наближено заданими вхідними даними на MIMD-комп'ютері // Проблемы программирования. – 2000. – № 1-2. – С. 102 – 112.