МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
им. М. В. ЛОМОНОСОВА

Научно-исследовательский вычислительный центр

О. Б. Арушанян, Н. И. Волченскова

СРЕДСТВА АВТОМАТИЗАЦИИ ДЛЯ РЕШЕНИЯ ЗАДАЧ ЛИНЕЙНОЙ АЛГЕБРЫ НА РАСПРЕДЕЛЕННОЙ ПАМЯТИ НА ОСНОВЕ ПАКЕТА ScaLAPACK

Учебное пособие

Москва, 2015

О.Б. Арушанян, Н.И. Волченскова

Средства автоматизации для решения задач линейной алгебры на распределенной памяти на основе пакета ScaLAPACK

(Учебное пособие)

Оглавление

1. Введение.

Настоящее учебное пособие составлено с целью помочь пользователям современных суперкомпьютеров с распределенной памятью добиться повышения производительности вычислений при решении задач линейной алгебры.

Речь идет о реализованном в НИВЦ МГУ (на основе пакета ScaLAPACK) комплексе параллельных программ по линейной алгебре PARALG (http://num-anal.srcc.msu.ru/par_prog/comparp.htm).

Этот комплекс содержит целевые программы, позволяющие на распределенной памяти решать системы линейных алгебраических уравнений и линейную проблему собственных значений. Кроме того, в комплекс включены средства автоматизации использования этих целевых программ для снижения трудозатрат и достижения наилучшей производительности при алгебраических вычислениях.

Для начала кратко поясним, что упомянутый выше комплекс параллельных программ по линейной алгебре PARALG, представляет собой русско-язычный аналог известного пакета ScaLAPACK [17, 18].

В его основе лежат те же принципы реализации параллельных алгоритмов, которые приняты в ScaLAPACKe, и при реализации целевых программ комплекса PARALG использовались базовые модули пакета ScaLAPACK. В комплексе PARALG содержатся и задокументированы (т.е. снабжены инструкциями по использованию) именно готовые целевые программы решения задач линейной алгебры на распределенной памяти.

Комплекс содержит большее количество целевых программ, чем в пакете ScaLAPACK (около шести десятков). Базовые же модули (как составные части целевых программ) вызываются непосредственно из библиотеки ScaLAPACKa.

Общие правила и принципы, приводимые в п. 2 настоящего пособия, в одинаковой мере справедливы как для пакета ScaLAPACK, так и для комплекса PARALG.

Использование готовых целевых программ комплекса PARALG значительно облегчает эффективное решение задач линейной алгебры. Тем не менее, у неспециалистов в данной области (как показывает опыт) это может вызвать определенные трудности, так как требует знакомства с недостаточно известными действиями и понятиями пакета ScaLAPACK, связанными с обеспечением эффективности параллельных вычислений. Сюда относятся такие, например, особенности параллельной реализации алгоритмов как:

— организация параллельных процессов в так называемую (виртуальную) решетку процессов и сопоставление с выбранной решеткой определенного целочисленного значения параметра, который однозначно характеризует выбранную решетку и называется указателем контекста или просто кон-

текстом;

— специальное распределение блоков матриц по параллельным процессам решетки;

— понятие и правильное определение так называемых дескрипторов матриц и связанное с этим представление о локальных и глобальных параметрах целевых программ комплекса PARALG.

Кроме того, требуется овладеть правильным использованием подпрограмм пакета BLACS (Basic Linear Algebra Communication Subprograms) [21, 22], входящего в состав пакета ScaLAPACK и выполняющего на более высоком уровне функции примитивов MPI [1, 2]. Указанные особенности тоже обсуждаются в п. 2 учебного пособия.

В п. 3 пособия рассматривается, реализованная в НИВЦ МГУ, надстроечная (системная) часть комплекса PARALG и правила ее использования для упрощения взаимодействия пользователей с целевыми программами.

В п. 4 рассматривается еще одно, реализованное в НИВЦ МГУ, средство автоматизации, позволяющее не только снизить трудозатраты пользователей, но и повысить производительность проводимых вычислений. Это — подсистема автоматизированного подбора наилучших (с точки зрения производительности) параметров распараллеливания вычислений.

В п. 5 разбираются примеры по решению алгебраических задач на распределенной памяти с использованием средств автоматизации комплекса PARALG.

В п. 6 представлен общий список входящих в настоящий момент в комплекс надстроечных программ автоматизации вычислений.

В п. 7 приводится набор команд, которые необходимо использовать при запуске рассматриваемых задач в ОС Linux и при получении результатов в процессе работы пользователей комплекса PARALG на суперкомпьютере “Чебышев”.

Работа по реализации комплекса PARALG выполнялась на суперкомпьютере “Чебышев”.

2. Комплекс параллельных программ PARALG (общее описание).

2.1. Организация и структура комплекса PARALG и его функциональное наполнение.

Рассматриваемый Комплекс программ относится к разряду “параллельных предметных библиотек”, при обращении к которым пользователю не приходится явно применять какие-либо конструкции специальных языков параллельного программирования или интерфейсов, поддерживающих взаимодействие параллельных процессов. Все необходимые действия по совместной работе па-

раллельных процессов (при решении задач линейной алгебры) выполняются подпрограммами из пакетов с именами PBLAS [19, 20] и BLACS (Basic Linear Algebra Communication Subprogramms), являющихся составными частями пакета ScaLAPACK.

Пакет BLACS является библиотекой подпрограмм, предназначенных для использования методов распараллеливания при решении задач линейной алгебры. Она разработана для компьютеров с распределенной памятью и обеспечивает запуск параллельных процессов и их взаимодействие посредством обмена сообщениями. При этом BLACS, являясь надстройкой над комплексом стандартных примитивов MPI, предназначенных для взаимодействия параллельных процессов посредством обмена сообщениями (Massage Passing Interface), освобождает пользователей от необходимости их изучения.

Другим пакетом, к которому обращаются программы комплекса, является пакет PBLAS (Parallel Basic Linear Algebra Subprogramms), содержащий версии для распределенной памяти подпрограмм (первого, второго и третьего уровней) хорошо известного пакета BLAS (Basic Linear Algebra Subprogram) [24, 25]. В пакете BLAS содержатся версии подпрограмм, реализующих базовые операции линейной алгебры (такие, как действия над векторами и скалярами, умножение матрицы на вектор или перемножение двух матриц). Разработчики пакета PBLAS реализовали его таким образом, чтобы интерфейс его подпрограмм был максимально похожим на интерфейс подпрограмм пакета BLAS.

Таким образом целевые программы описываемого комплекса (т.е. программы, которые имеют самостоятельное значение при решении прикладных задач) могут содержать обращения к подпрограммам указанных выше пакетов BLACS, PBLAS и BLAS, а также к базовым подпрограммам пакета ScaLAPACK.

К базовым подпрограммам пакета ScaLAPACK относятся подпрограммы, которые не используются независимо от других программ Комплекса. Они всегда выступают в роли вызываемых из других подпрограмм. Например, подпрограмма умножения матрицы общего вида на ортогональную матрицу, полученную в результате QR-разложения при помощи другой подпрограммы.

Самую важную для пользователя часть функционального наполнения комплекса PARALG составляют целевые программы, которых к настоящему времени насчитывается около 60. Они предназначены для решения задач из таких разделов линейной алгебры, как решение систем линейных алгебраических уравнений, решение линейной и обобщенной проблемы собственных значений и сингулярное разложение матриц. При этом разные программы используются для матриц разных видов.

Помимо целевых программ, для комплекса PARALG была разработана и реализована программная надстройка, которая избавляет пользователей, не яв-

ляющихся специалистами, от лишних трудозатрат при работе с целевыми программами комплекса (см. п. 3).

2.2. Специфика параллельных алгоритмов и использование пакета BLACS для организации параллельных процессов.

Для того чтобы лучше понять смысл использования надстроечных подпрограмм, рассмотрим вкратце специфику используемых в комплексе параллельных алгоритмов.

Главным объектом преобразований при решении задач линейной алгебры являются матрицы. Для их эффективной обработки с использованием параллельных процессов создателями пакета ScaLAPACK были реализованы так называемые блочные алгоритмы (или блочные версии известных алгоритмов линейной алгебры). Они предполагают разбиение обрабатываемых матриц на блоки с последующим равномерным распределением этих блоков по определенным правилам по всем используемым параллельным процессам для одновременной обработки каждым из них своей части исходной матрицы (подробнее см. п. 2.3). Такое распределение обеспечивает не только сокращение времени на обработку всей матрицы, но и достаточно равномерную загрузку всех процессов.

Эти алгоритмы основаны на представлении о так называемой "решетке процессов". Эта логическая конструкция организует параллельные процессы в двумерную структуру, состоящую из R строк и C столбцов. При такой организации и принятых правилах распределения все части одной строки распределенной матрицы будут располагаться в процессах, относящихся к одной и той же строке решетки процессов, а части столбца матрицы — в процессах одного и того же столбца решетки процессов.

Подпрограммы пакета BLACS как раз и разработаны специально для обеспечения эффективной обработки матриц, распределенных по решетке процессов.

В принятой в пакете BLACS логической структуре параллельных процессов каждый процесс идентифицируется с помощью пары чисел — координат процесса в решетке процессов.

На рисунке ниже изображена решетка из 8 процессов, состоящая из двух строк и четырех столбцов (2×4).

	0	1	2	3
0				
1				

Обозначаются эти процессы как имеющие следующие координаты:
$(0,0)$, $(0,1)$, $(0,2)$, $(0,3)$, $(1,0)$, $(1,1)$, $(1,2)$, $(1,3)$.

Иными словами, в общем виде, если в решетке процессов R строк и C столбцов, любой процесс в решетке обозначается парой (i, j), где $0 \leqslant i < \mathrm{R}$, $0 \leqslant j < \mathrm{C}$. Общее число процессов равно $\mathrm{R} * \mathrm{C}$.

В компьютерах с распределенной памятью каждый параллельный процесс работает со своей локальной памятью, а обмен информацией между процессами осуществляется посредством передачи сообщений через связывающую их сеть.

Пакет BLACS, предназначенный для таких компьютеров, опирается на использование широко известных и распространенных интерфейсов обмена сообщениями MPI (Massage Passing Interface) или PVM (Parallel Virtual Machine). Они содержат не только операции обмена между одним посылающим и одним принимающим сообщение процессом, но также и массовые операции обмена, когда, например, один процесс может рассылать сообщение сразу всем другим параллельным процессам, и, наоборот, один из процессов может получать сообщения от всех других.

В пакете BLACS такие операции называются контекстными операциями (scope operations), а про процессы, принимающие в них участие, говорят, что они находятся внутри операционного контекста (operation's scope). Организация процессов в виде логической решетки позволяет пакету BLACS естественным образом расширить виды контекстов, а именно, объявлять (задавать) в качестве контекста, т.е. области действия массовых операций обмена сообщениями, не только **все** параллельные процессы, но также и все процессы, принадлежащие только некоторой строке решетки процессов или некоторому ее столбцу. Это обеспечивает программисту линейной алгебры более удобные и гибкие возможности распараллеливания.

Использование понятия решеток процессов может служить не только удобству программирования, но и повышению эффективности (скорости) вычислительного процесса. Однако это произойдет только в том случае, если в базовой машине параллельные процессы связаны между собой с помощью, по крайней мере, двумерных сетей. В большинстве современных суперкомпьютеров это условие выполнено, но при использовании сети Ethernet такое преимущество недостижимо.

2.3. Алгоритм распределения матриц по параллельным процессам.

Рассмотрим более подробно наиболее общий алгоритм распределения разбитой на блоки прямоугольной матрицы по параллельным процессам. Целевые программы комплекса PARALG (как и базовые программы ScaLAPACKa) ориентированы на конкретный способ распределения блоков матрицы по решетке параллельных процессов, называемый блочно-циклическим.

Пусть прямоугольная матрица размера $\mathrm{M} \times \mathrm{N}$ разбивается на прямоуголь-

ные блоки размера MB $\times$ NB начиная с левого верхнего угла матрицы. Все горизонтально примыкающие друг к другу блоки будут равномерно и циклически распределяться между процессами одной и той же строки решетки. Это происходит аналогично тому, как сдаются карты участникам игры перед ее началом — по кругу: после раздачи всем по одной карте, раздача по второй карте опять начинается с первого игрока и т.д.

Точно так же все вертикально примыкающие друг к другу блоки матрицы будут равномерно и циклически распределены между всеми процессами одного и того же столбца решетки процессов. Таким образом, все распределенные на какой-либо процесс блоки, будучи "прижаты" в памяти друг к другу по горизонтали и вертикали, составят единый двумерный массив, являющийся локальной частью исходной глобальной матрицы.

Рассмотрим более строго, каким же образом блоки, распределенные на один и тот же процесс, хранятся в локальной памяти этого процесса. Другими словами, мы выпишем точные формулы, которые связывают элемент глобальной матрицы, определяемый глобальными индексами I и J (т.е. A (I, J)), с координатами процесса, владеющего этим элементом, в решетке процессов (Pr, Pc) и с расположением этого элемента матрицы в локальной памяти этого процесса.

Рассмотрим сначала, для простоты, эти связи на примере одномерного случая, т.е. размещение одномерного глобального массива длины N, разбитого на блоки длины NB, по одномерной решетке из P процессов, перенумерованных начиная от 0 до (P - 1). Элементы самого глобального массива нумеруются от 1 до N. Прежде всего, массив делится на смежные блоки размера NB. Когда N не делится на NB нацело, последний блок массива элементов будет содержать только mod (N, NB) элементов вместо NB. Блоки, на которые разбивается исходный массив, нумеруются (как и процессы), начиная с 0. Они распределяются, как уже говорилось, по кругу (т.е. циклически).

Другими словами, мы предполагаем, что процесс с номером 0 получает первый блок (с номером 0), k-й блок связывается с процессом, имеющим координату (номер) mod (k, P). Блоки, связанные с одним и тем же процессом, хранятся в памяти рядом (в смежных, прилегающих частях). Отображение элемента массива с глобальным индексом I определяется следующим соотношением:

$$I = k * NB + x = (m * P + p) * NB + x,$$

где

I — глобальный индекс элемента глобального массива;

m — локальная координата блока, в котором этот элемент расположен;

p — координата процесса, владеющего этим блоком;

x — локальная координата элемента внутри того блока, где находится элемент глобального массива с индексом I.

Легко установить соотношения между этими переменными:

$$\begin{aligned} \mathrm{p} &= \big[(\mathrm{I}-1)/\mathrm{NB}\big] \bmod \mathrm{P}, \\ \mathrm{m} &= \big[(\mathrm{I}-1)/(\mathrm{P} * \mathrm{NB})\big], \\ \mathrm{x} &= \bmod(\mathrm{I}-1,\, \mathrm{NB}) + 1. \end{aligned}$$

Эти уравнения позволяют определить локальную информацию (т.е. локальный индекс $\mathrm{m} * \mathrm{NB} + \mathrm{x}$), так же как и координату процесса p, соответствующую глобальному элементу, идентифицируемому его глобальным индексом I, и наоборот.

В таблице ниже показано отображение в локальную память при разбиении массива на блоки, когда $\mathrm{P} = 2$, $\mathrm{N} = 16$ и $\mathrm{NB} = 8$.

I	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
p	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
m∗NB+x	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8

В двумерном случае предположим, что матрица разделена на $\mathrm{MB} \times \mathrm{NB}$ блоки и что первый блок передан процессу с координатами $(\mathrm{RSRC}, \mathrm{CSRC})$. Формула, приведенная выше для одномерного случая, должна использоваться повторно независимо для каждого измерения решетки процессов $\mathrm{Pr} \times \mathrm{Pc}$. Например, элемент матрицы (I, J) находится в процессе с координатами $(\mathrm{Pr}, \mathrm{Pc})$ внутри локального блока (m, n) в позиции (x, y), задаваемой формулами

$$\begin{aligned} (\mathrm{m}, \mathrm{n}) &= \Big(\big[(\mathrm{I}-1)/(\mathrm{Pr} * \mathrm{MB})\big],\ \ \big[(\mathrm{J}-1)/(\mathrm{Pc} * \mathrm{NB})\big]\Big), \\ (\mathrm{Pr}, \mathrm{Pc}) &= \Big(\big(\mathrm{RSRC} + [(\mathrm{I}-1)/\mathrm{MB}]\big) \bmod \mathrm{Pr},\ \ \big(\mathrm{CSRC} + [(\mathrm{J}-1)/\mathrm{NB}]\big) \bmod \mathrm{Pc}\Big), \\ (\mathrm{x}, \mathrm{y}) &= \big(\bmod(\mathrm{I}-1, \mathrm{MB}) + 1,\ \ \bmod(\mathrm{J}-1, \mathrm{NB}) + 1\big). \end{aligned}$$

Эти формулы показывают, как матрица A размера $\mathrm{M\_A} \times \mathrm{N\_A}$ отображается и хранится на решетке процессов. Она сначала разбивается на $\mathrm{MB\_A} \times \mathrm{NB\_A}$

блоки начиная с ее верхнего левого угла. Эти блоки затем равномерно распределяются по решетке процессов циклическим образом.

Каждый процесс владеет набором блоков, которые хранятся рядом (смежно) по столбцам в двумерном массиве, расположенном в памяти "по столбцам".

Это соглашение о локальном размещении позволяет эффективно использовать иерархию локальной памяти посредством вызова пакета BLAS для подмассивов, которые могут быть больше, чем один блок размера MB_A × NB_A.

На рисунке ниже представлено отображение матрицы 5×5, разделенной на блоки размером 2×2 на решетку процессов размером 2×2 (т.е. M_A = N_A = 5, Pr = Pc = 2 и MB_A = NB_A = 2). Локальные элементы каждого столбца матрицы хранятся рядом в памяти каждого процесса.

a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}
a_{31}	a_{32}	a_{33}	a_{34}	a_{35}
a_{41}	a_{42}	a_{43}	a_{44}	a_{45}
a_{51}	a_{52}	a_{53}	a_{54}	a_{55}

	0			1	
0	a_{11}	a_{12}	a_{15}	a_{13}	a_{14}
	a_{21}	a_{22}	a_{25}	a_{23}	a_{24}
	a_{51}	a_{52}	a_{55}	a_{53}	a_{54}
1	a_{31}	a_{32}	a_{35}	a_{33}	a_{34}
	a_{41}	a_{42}	a_{45}	a_{43}	a_{44}

Цифры 0 и 1 в левой и верхней части последней таблицы означают номера строк и столбцов решетки процессов соответственно.

Важной задачей для пользователя является определение числа строк и столбцов плотной глобальной матрицы, которое получает каждый конкретный процесс. Для выполнения этой функции существует служебная подпрограмма NUMROC.

Для локального числа строк используется обозначение LOCr ( ), а для локального числа столбцов — LOCc ( ). Значения LOCr ( ) и LOCc ( ), получаемые подпрограммой NUMROC, являются результатом точных вычислений.

Однако если требуется понять общую идею вычисления размера локального массива, то можно проделать следующее грубое вычисление верхней границы этой величины:

а) верхняя граница для LOCr ( ) оценивается по формуле

$$\mathrm{LOCr}\,(\ ) = \frac{\frac{\mathrm{M\_A} + \mathrm{MB\_A} - 1}{\mathrm{MB\_A}} + \mathrm{Pr} - 1}{\mathrm{Pr}} * \mathrm{MB\_A}$$

или

$$\mathrm{LOCr}\,(\ ) = \big[[\mathrm{M\_A}/\mathrm{MB\_A}]/\mathrm{Pr}\big] * \mathrm{MB\_A};$$

б) величина LOCc ( ) оценивается по формуле

$$\mathrm{LOCc}\,(\ ) = \frac{\frac{\mathrm{N\_A} + \mathrm{NB\_A} - 1}{\mathrm{NB\_A}} + \mathrm{Pc} - 1}{\mathrm{Pc}} * \mathrm{NB\_A}$$

или

$$\mathrm{LOCc}\,(\ ) = \big[[\mathrm{N\_A}/\mathrm{NB\_A}]/\mathrm{Pc}\big] * \mathrm{NB\_A}.$$

Заметим, что эти вычисления могут привести к очень большой переоценке величины действительно требующегося пространства.

В комплексе PARALG (как и в ScaLAPACKe) предполагается, что такое блочно–циклическое распределение матрицы по решетке процессов будет произведено пользователем перед его обращением к целевой программе решения задачи.

Ясно, что такое “замысловатое” распределение исходной матрицы в локальную память каждого из параллельных процессов “вручную” (т.е. посредством использования операторов присваивания) может быть относительно легко выполнено пользователем только в случае очень небольших матриц.

В учебном пособии [14] (в Приложениях 2 и 3) рассмотрены два других, более автоматизированных способа распределения матриц: с помощью служебной подпрограммы PDELSET (удобно, если матрица имеет какую-то регулярную структуру) или с помощью подпрограммы PDLAREAD, которая, читая из файла хранящиеся там значения элементов матрицы, распределяет их по параллельным процессам в соответствии с описанным общим правилом.

2.4. Параметры целевых программ комплекса и дескрипторы глобальных массивов.

Программы Комплекса составлены на языке ФОРТРАН-77 в стиле SMPD (Single Program Multiple Data). Это означает, что одна и та же исполняемая программа загружается во все параллельные процессы, заказанные для решения задачи в команде запуска на счет. В то же время, каждый процесс занимается обработкой своих данных, составляющих некоторую часть общих данных задачи. Эти данные отдельного процесса (их называют локальными данными)

непосредственно недоступны другим процессам и хранятся в отдельной (локальной) памяти каждого процесса. Необходимый обмен данными между разными процессами производится только посредством передачи сообщений (технология MPI [1, 2, 10–13]).

Из сказанного следует, что исходную матрицу A не требуется хранить целиком в оперативной памяти процесса как единый массив. На практике такие матрицы (векторы) в виде единого целого могут храниться во внешних файлах (на дисковой памяти). По этой причине такую матрицу A называют **глобальной** матрицей. Поскольку перед началом работы целевых параллельных программ комплекса исходные глобальные матрицы (или векторы) разделяются на части и распределяются по параллельным процессам, то в дальнейшем нередко употребляется термин "распределенная глобальная матрица (вектор)". Часть глобальной матрицы (вектора), представленная в локальной памяти одного из параллельных процессов, называется **локальной** частью матрицы (вектора). Совокупность всех локальных частей на всех, используемых при решении задачи параллельных процессах, и составляет **распределенную глобальную** матрицу (вектор).

Поскольку каждый процесс работает со своей локальной частью данных (в нашем случае фрагментом глобальной матрицы), то в головной фортранной программе достаточно выделить (заказать) память, необходимую только для максимальной локальной части этой матрицы из всех локальных частей, обрабатываемых на параллельных процессах.

Локальные части, обрабатываемые на каждом из процессов, могут быть как одинакового, так и разного размера. Для того чтобы каждый из процессов мог разместить свою локальную часть данных (матрицы), заказываемая область локальной памяти должна быть не меньше максимальной локальной части данных. Величины локальных частей распределенной матрицы могут различаться на разных процессах из-за того, что при блочно-циклическом распределении матрицы некоторым процессам может достаться меньшее количество блоков, чем другим.

Само блочно-циклическое распределение матрицы по параллельным процессам не определяет однозначно, в память какого именно процесса попадет тот или иной элемент глобальной матрицы. Для одной и той же матрицы это зависит как от общего количества используемых параллельных процессов (и количества процессов в строках и столбцах решетки), так и от величины блоков, на которые разбивается глобальная матрица.

Для сохранения информации о конкретном способе разбиения глобального объекта на блоки и размещении блоков в локальной памяти всех процессов вводится составной объект, называемый **дескриптором** глобального массива.

Дескриптор представляет собой одномерный массив, состоящий из 9 или 7 элементов (в зависимости от типа дескриптора). В дескрипторе хранится информация о размерах глобального массива (число строк и столбцов матрицы), о размерах блоков, на которые она разбивается, о номере процесса, в память которого распределяется первый блок глобальной матрицы (левый верхний угол), а также некоторая другая информация. Подробнее о дескрипторах см. ниже.

По информации в дескрипторах, а также учитывая общие для комплекса правила распределения блоков матриц (векторов) по параллельным процессам, всегда можно определить, в локальной памяти какого процесса и где именно находится элемент глобальной матрицы с глобальными индексами i и j (см. подробнее в п. 2.3).

Организация решетки процессов, обмен информацией между процессами, а также другие действия, связанные с процессами, выполняются с помощью подпрограмм пакета BLACS (см. п. 2.2). Пользователю, однако, в большинстве случаев не требуется знание всех возможностей этого пакета. При решении одной из задач линейной алгебры он должен сделать в головной программе лишь несколько стандартных обращений к нескольким подпрограммам пакета BLACS. Наглядные примеры этого можно найти, например, в предлагаемом пользователям описании целевой параллельной программы комплекса (в разделе "Пример использования", см. Приложение 1 в учебном пособии [14]), а также в фортранном тексте тестового примера к этой подпрограмме (см. также Приложение 2).

Из сказанного выше вытекает, что формальные параметры целевых программ комплекса могут быть глобальными или локальными. Глобальными называются параметры подпрограмм, которые относятся к указанным выше глобальным объектам. Например, глобальными являются число строк или столбцов исходной матрицы A (M, N) или число наддиагоналей (BWU) в матрице A, если она является ленточной. Локальными называются параметры, которые характеризуют объекты, определяемые в локальной памяти отдельного процесса. Например, указатель на локальную память, занимаемую локальной частью глобальной матрицы A или глобального вектора правых частей системы уравнений B.

Некоторые затруднения у пользователей при обращении к целевым программам комплекса вызывает задание такого обязательного составного фактического параметра, как дескриптор массива (матрицы).

Уже упоминавшиеся выше дескрипторы массивов характеризуются одновременно и как глобальные, и как локальные параметры подпрограмм, поскольку среди их элементов, определяющих в основном характеристики глобальных массивов, есть характеристика, описывающая размещение части глобального

массива в локальной памяти процесса (ведущая размерность локального массива).

Каждому глобальному массиву (матрице или вектору), который необходимо распределить по параллельным процессам, следует сопоставить специальный объект, называемый дескриптором. Дескриптор представляет собой одномерный массив, состоящий из 9 или 7 элементов целого типа (в смысле ФОРТРАНа). Он предназначен для хранения информации, конкретизирующей способ разбиения глобального массива на блоки и их распределение по всем параллельным процессам.

Дескрипторы принято обозначать идентификатором DESC с добавлением в конце имени массива (матрицы или вектора). Например, DESCA означает дескриптор матрицы A, DESCB — дескриптор вектора B. Дескрипторы бывают трех типов, которые обозначаются целыми числами, равными 1, 501 и 502.

Для плотных глобальных матриц A, которые распределяются по двумерной решетке процессов, используется дескриптор, имеющий тип 1 и состоящий из 9 элементов, за каждым из которых закреплено свое символическое имя (см. таблицу ниже). В таблице указываются номер (индекс) каждого элемента в массиве, его символическое имя (которое оканчивается символом подчеркивания "_"), а также его смысл. Идентификатор, стоящий в символическом имени элемента после символа подчеркивания "_", является идентификатором глобального массива, описываемого данным дескриптором. Например, N_A обозначает элемент дескриптора, который содержит число столбцов глобальной матрицы A.

Таблица элементов дескриптора плотной глобальной матрицы

N	**Имя**		**Смысл**
1.	DTYPE_A	—	тип дескриптора (для плотной матрицы A DTYPE_A = 1)
2.	CTXT_A	—	обозначение контекста BLACS'а, соответствующего выбранной решетке процессов, по которой распределяется глобальная матрица A. Целое значение, устанавливаемое подпрограммой из пакета BLACS. См. п. 2.2
3.	M_A	—	число строк в глобальной матрице A
4.	N_A	—	число столбцов в глобальной матрице A
5.	MB_A	—	число строк в блоках, на которые разбивается глобальная матрица A
6.	NB_A	—	число столбцов в блоках, на которые разбивается глобальная матрица A

7. RSRC_A — номер строки процесса в решетке процессов, куда был распределен первый элемент глобальной матрицы A (левый верхний угол); как правило, удобнее всего распределять первый элемент в процесс с номером $(0,0)$
8. CSRC_A — номер столбца процесса в решетке процессов, куда был распределен первый элемент глобальной матрицы A (левый верхний угол); как правило удобнее всего распределять первый элемент в процесс с номером $(0,0)$
9. LLD_A — ведущая размерность локального массива (LLD_A $\geqslant$ MAX(1, LOCr(M_A)).

Например, записи DESCA(3) и DESCA(M_A) означают один и тот же элемент дескриптора матрицы A, содержащий число строк этой матрицы.

2.5. Параметры распараллеливания, трудности выбора, возможные направления автоматизации.

Если проанализировать список входных параметров целевых программ комплекса PARALG, то помимо разделения их на глобальные и локальные (что описывалось в предыдущем п. 2.4), их можно разделить на две группы по смысловому предназначению. Часть параметров определяется математической постановкой задачи и ее конкретизирует, т.е. влияет на то, какие именно численные результаты будут получены. Это, например, параметры, задающие исходную матрицу или вектор правой части системы уравнений и т.п.

Другая же часть параметров определяет то, каким образом вычислительная нагрузка будет распределена между параллельными процессами. Назовем такие параметры "параметрами распараллеливания" вычислений. Их выбор влияет не на значения получаемых результатов решения задачи, а на время, за которое эти результаты будут получены (т.е. на производительность вычислений) при той же самой постановке задачи.

Сюда относятся такие параметры, как:

— количество строк и столбцов в решетке процессов (NPROW и NPCOL, см. также п. 2.2);

— величина блоков, на которые разбиваются матрицы перед распределением по параллельным процессам (NB) (без сколько-нибудь существенного умаления общности, вместо прямоугольных блоков MB×NB будем в дальнейшем рассматривать квадратные блоки NB×NB).

Выбор этих параметров в комплексе PARALG (как и в ScaLAPACKe) первоначально возлагался на пользователя целевой программы. Однако выбор достаточно приемлемых и тем более наилучших (с точки зрения производитель-

ности) значений этих параметров — задача нетривиальная. Она является затруднительной для пользователей, не являющихся специалистами в рассматриваемой области.

Поэтому разработчиками комплекса PARALG было принято решение реализовать для разных разделов линейной алгебры набор подпрограмм, выдающих по запросу пользователя, перед обращением к целевой программе, некоторый приемлемый набор значений параметров распараллеливания для конкретной задачи.

Удобство использования этих подпрограмм выбора параметров распараллеливания состоит еще и в том, что они являются обычными последовательными программами, не требуют заказа нескольких параллельных процессов и используют мизерное время для выдачи результатов. Кроме того, они выдают значение еще одной величины, зависящей от указанных выше параметров распараллеливания, которую пользователям необходимо передать при обращении к целевой программе. Это — размер локальной памяти, которую необходимо выделить на каждом из параллельных процессов при решении задачи.

Таким образом подпрограммы выбора параметров "закрывают" (берут на себя) одно из направлений возможной автоматизации (одну из "неприятных" обязанностей пользователя) при обращении к целевым программам.

Другим направлением работ, которое, как показывает опыт, требует от пользователя значительных трудозатрат, являются необходимые подготовительные работы перед обращением к целевой программе, само обращение к ней и завершающие действия по выводу полученных результатов.

Разработчиками комплекса PARALG также был реализован набор параллельных подпрограмм для разных разделов линейной алгебры, которые позволяют автоматизировать упомянутый комплекс работ. Более подробно оба набора подпрограмм автоматизации выбора параметров и автоматизации вызова целевых программ рассматриваются в п.п. 3.1 и 3.2 настоящего пособия.

Однако, пожалуй, наиболее сложным при решении прикладных задач является вопрос, связанный с выбором наилучших параметров распараллеливания при заданном количестве параллельных процессов, который позволяет добиться при этом наилучшей производительности вычислений в каждом конкретном случае.

Теоретически обоснованный подбор наилучших параметров является весьма сложной задачей (что признается самими разработчиками ScaLAPACKa), решение которой зависит от целого ряда факторов (причин):

— конкретной программы комплекса, используемой для расчетов (прежде всего, от алгоритмов, реализованных в базовых подпрограммах);

— от характеристик конкретной вычислительной системы (в частности, от характеристик используемой сети взаимосвязи процессоров);

— от особенностей реализации на данной вычислительной системе пакета BLACS, который обеспечивает обмен данными между процессорами;

— а также от размера задачи (порядка матриц).

Рассмотрим некоторые соображения (замечания), относящиеся к выбору параметров решетки процессов NPROW и NPCOL.

Так, например, LU–, QR– и QL–разложения матриц лучше (быстрее) выполняются на решетках процессов "плоской" прямоугольной формы, у которых число строк в решетке меньше числа столбцов (Pr $<$ Pc).

В то же время, LQ– и RQ–разложения быстрее выполняются на "высоких" ("вертикальных") решетках процессов, у которых число строк больше числа столбцов (Pr $>$ Pc).

Квадратные или близкие к квадратным решетки (Pr $\approx$ Pc) являются более предпочтительными, например, для алгоритмов обращения матриц или операций приведения (преобразования) матриц к двухдиагональной форме, трехдиагональной форме или форме Хессенберга.

Однако результаты анализа алгоритмов и советы по выбору предпочтительной формы решетки в той или иной степени могут зависеть от физических характеристик используемой сети взаимосвязи процессоров. Когда пользователю неизвестны особенности реализации пакета BLACS и свойства используемой сети для обмена данными на конкретной вычислительной системе, еще одно соображение может помочь с выбором формы решетки процессов.

На задачах малого размера на производительность бóльшее влияние оказывает общее число обменов данными (передаваемых сообщений), тогда как для задач среднего размера определяющим фактором становится общий объем передаваемых данных. Для задач большого размера основное влияние на производительность оказывает выполнение операций с плавающей запятой.

При использовании одномерной решетки процессов (Pr = 1, Pc = NPROC) сокращается общее число обменов данными в сети связи, но увеличивается общий объем передаваемых сообщений. Поэтому одномерное распределение данных лучше использовать для задач малых размеров, но хуже для задач больших размеров (особенно, когда при этом используется более 8 процессоров).

Поскольку целевая программа может обращаться не к одной базовой подпрограмме, а свойства конкретной сети обмена данными и реализации пакета BLACS неизвестны или их трудно учесть, то (теоретический) выбор наилучшей формы решетки процессов становится "неподъемной" задачей (особенно для простого пользователя-практика).

Что же касается выбора наилучшего (для производительности) размера блока, на которые разбиваются матрицы, то он также может зависеть от конкретной вычислительной системы, и обычно подбор делается эмпирическим путем.

В связи с вышесказанным, авторы комплекса PARALG пришли к выводу о полезности реализации (в его рамках) некоторой подсистемы автоматизированного экспериментального подбора наилучших параметров распараллеливания задачи. Реализация данной подсистемы автоматизации описана в п. 4.

В п. 5 описывается порядок действий пользователя, решающего задачу с использованием средств автоматизации.

3. Надстроечная системная часть комплекса PARALG для автоматизации доступа к целевым программам комплекса.

3.1. Подпрограммы автоматизации выбора параметров распараллеливания для целевых программ.

В п. 2.5 уже кратко рассказывалось назначение этих подпрограмм, называемых подпрограммами выбора параметров. В настоящее время в комплекс включено около десятка таких подпрограмм для разных подразделов систематического каталога комплекса и разных видов матриц (полный их список см. в п. 6 настоящего пособия).

При обращении к этим подпрограммам пользователю достаточно задать только параметры, определяющие постановку задачи и выделяемое общее число параллельных процессов, а именно:

- имя выбранной для расчетов целевой подпрограммы комплекса (NAME);
- количество NPROCS процессов, которые могут быть выделены для решения задачи;
- размеры исходных(ой) матриц(ы) (M — число строк матрицы, N — число столбцов (или порядок N квадратной) матрицы и, в случае ленточной матрицы, BWL и BWU — число под- и наддиагоналей).

В результате работы такой подпрограммы пользователь получит некоторый приемлемый набор (возможно не наилучший) значений параметров распараллеливания, а именно:

- размеры блока (клетки), на которые будут разделены исходные матрицы и исходный вектор правой части (в случае решения систем уравнений) (NB);
- размеры решетки процессов, которая будет использоваться при вычислениях (NPROW — число строк в решетке, NPCOL — число столбцов в решетке), тем самым определяя и общее число процессов, которое будет реально

использоваться при расчетах NPROW $*$ NPCOL (оно может быть меньше заданного NPROCS);

- минимальный размер локальной памяти, которую необходимо будет выделить на каждом из процессов (MMIN).

Величина MMIN зависит в том числе и от указанных выше других выходных величин (NB, NPROW, NPCOL). Но в первую очередь она зависит от вычислительных алгоритмов, используемых выбранной целевой программой, а также от размеров задачи (матриц). Значение MMIN пользователю определить значительно труднее, чем первых трех.

При этом подпрограммы выбора реализованы так, что параметр NB (величина квадратного блока матрицы) является как входным, так и выходным. Это сделано для того, чтобы более опытный пользователь мог сам выбрать и указать величину блока. Тогда подпрограмма выбора выдаст, с учетом этого NB, соответствующие приемлемые значения остальных параметров распараллеливания. Пользователь, который доверяет выбор значения NB подпрограмме, должен в этом случае подать ей на вход NB $= 0$. Пользователь, желающий самостоятельно установить величину блока, может выбрать ее, вообще говоря, любой в пределах $1 <$ NB $<$ N (где N — порядок глобальной матрицы). Хотя ясно, что значения, близкие к крайним, не улучшат производительности.

Кроме того, по желанию, пользователь может выбрать самостоятельно и один из размеров решетки процессов, положив его, вместо 0, равным конкретному значению ($1 \leqslant$ NPROW $\leqslant$ NPROCS или $1 \leqslant$ NPCOL $\leqslant$ NPROCS).

По желанию пользователя подпрограммы вычисления параметров могут выдать ему значение всех параметров распараллеливания, включая минимальный размер локальной памяти (NPROW, NPCOL, NB и MMIN).

Приведем типичный вид фортранного оператора вызова одной из таких подпрограмм (для целевых программ решения систем уравнений с матрицами общего вида):

CALL PAR_GESV(NAME, NPROCS, N, NPROW, NPCOL, NB, MMIN)

Здесь первые три входные параметра задаются пользователем (имя целевой программы, число выделяемых процессоров и порядок исходной матрицы).

Следующие четыре параметра — выходные (число строк и столбцов в решетке процессов, размер блока матрицы и минимальный размер локальной памяти).

Мизерность затрат машинного времени на работу такой подпрограммы выбора позволяет пользователю, которого не устраивает, например, выданное значение необходимой локальной памяти (MMIN), повторить и, возможно неодно-

кратно, расчет с другими входными параметрами до получения устраивающего результата.

3.2. Подпрограммы автоматизации подготовки входных данных, вызова целевых программ и выдачи результатов расчетов.

Эти подпрограммы дают возможность упростить для пользователя следующий шаг — сам запуск процесса параллельного решения задачи и выдачи результатов.

Эти подпрограммы (будем называть их далее "подпрограммами вызова" целевых подпрограмм) предполагают, что исходные матрицы и векторы хранятся и считываются из файлов, хранимых на внешней памяти.

При этом в начале файла должны быть записаны два целых числа, обозначающих число строк (M) и число столбцов (N) в следующей далее матрице (векторе). Идущие далее элементы матрицы должны быть упорядочены по столбцам (как это принято в языке ФОРТРАН).

Предполагается, что каждая матрица или вектор хранится в отдельном файле указанной структуры.

Эти подпрограммы вызова выполняют за пользователя чтение исходных (глобальных) матриц (и векторов) из внешних файлов и правильное распределение их частей по параллельным процессам (как этого требуют используемые алгоритмы, см. п. 2.3).

Следующий шаг, который выполняют подпрограммы вызова, — это формирование правильного списка фактических параметров и обращение с этим списком к выбранной пользователем целевой программе (параметр NAME).

Среди указанного списка параметров у пользователей наибольшее затруднение вызывают параметры, называемые дескрипторами глобальных матриц (см. п. 2.4). Таким образом, подпрограммы вызова и в этом смысле уменьшают их трудозатраты.

После передачи управления целевой программе, окончания счета по ней и получения результатов, чаще всего в виде матриц или векторов, распределенных по параллельным процессам, подпрограммы вызова формируют из распределенных результатов единые выходные глобальные массивы, записывая их во внешние файлы и/или выдавая на печать.

Обращения к самим подпрограммам вызова не являются затруднительными для пользователей, так как параметры распараллеливания уже были определены с помощью подпрограмм выбора (п. 3.1).

Приведем один из типичных примеров обращения к подпрограмме автоматизации вызова (для тех же целевых программ, что и в п. 3.1):

```
CALL CAL_GESV(NAME, N, NPROW,NPCOL, AMEM,MEMSIZE, NB, INFILEA,
```

```
            INFILEB,OUTFILE)

  Здесь все параметры - входные. Среди них можно выделить параметры,
 определяющие постановку задачи:

          NAME - имя целевой программы;
             N - порядок матриц(ы) (или вектора);
INFILEA,INFILEB - имена файлов, содержащих входные матрицы
                  (или векторы).

 Кроме того, параметры, определяющие условия распараллеливания,
 которые были получены с помощью соответствующей подпрограммы
 вычисления параметров (PAR_GESV):

   NPROW,NPCOL - число строк и столбцов в решетке процессов;
            NB - размер блока матрицы;
       MEMSIZE - размер массива AMEM выделяемой локальной памяти,
                 который не может быть меньше, чем значение MMIN,
                 полученное подпрограммой PAR_GESV.

 Наконец, последний параметр OUTFILE задает имя файла, в который
 будут записаны полученные результаты решения задачи.
```

В настоящее время реализовано и включено в комплекс около десятка таких подпрограмм (полный их список см. в п. 6 настоящего пособия) для разных подразделов систематического каталога комплекса и разных видов матриц.

4. Подсистема автоматизированного подбора наилучших параметров распараллеливания.

Выше в п. 2.5 упоминались соображения, указывающие на трудности теоретического подбора наилучших параметров распараллеливания при использовании целевых программ комплекса PARALG для решения задач линейной алгебры.

Включенная в комплекс подсистема автоматизированного экспериментального подбора наилучших параметров распараллеливания BSTPAR позволяет выполнить сразу (за один "подход") несколько просчетов для решения одной и той же конкретной задачи при использовании одной и той же целевой программы и одного и того же числа параллельных процессов. При этом каждый просчет производится со своим набором параметров распараллеливания (см. п.п. 2.5 и 3.1). В завершение подсистема BSTPAR по результатам сравнения всех

просчетов (данного “подхода”) выдает минимальное из всех время просчета и соответствующий ему (наилучший) набор параметров распараллеливания.

Такая подсистема позволяет выбрать один раз для некоторой целевой программы (т.е. для решения задачи линейной алгебры определенного вида с матрицей конкретного определенного порядка) и при использовании конкретного определенного числа параллельных процессов наилучший набор параметров распараллеливания. Выбранный набор параметров затем можно использовать неоднократно при необходимости решения аналогичных задач с разными матрицами того же порядка.

Таким образом, эта подсистема BSTPAR позволяет не только автоматизированным образом выбрать наилучшие параметры распараллеливания, но и сократить трудозатраты и сэкономить машинное время при неоднократном решении аналогичных задач.

Такие неоднократные расчеты с разными матрицами одного порядка могут потребоваться, например, пользователю-практику, а также при решении аналогичных задач разными пользователями.

Рассмотрим несколько подробнее устройство указанной подсистемы.

4.1. В ней можно выделить программную часть и архив результатов. Программная часть, в свою очередь, состоит из головной подпрограммы (BSTPAR9) и целого набора подпрограмм для каждого из функциональных подразделов комплекса PARALG (их список см. в п. 6). Будем в дальнейшем для простоты называть эти подпрограммы “подпрограммами одного просчета”. В задачу каждой из подпрограмм этого набора входит выполнение следующих главных функций:

— выполнить распределение по параллельным процессам заданных исходных глобальных матриц в соответствии с установленными общими правилами, а также с учетом конкретного набора параметров распараллеливания, полученного от головной подпрограммы подсистемы;

— построив правильный список фактических параметров, выполнить обращение к заданной пользователем целевой программе, инициализировав тем самым решение конкретной задачи с матрицей заданного порядка;

— зафиксировать с помощью специальных системных подпрограмм процессорное время, затраченное на получение результатов решения задачи, и передать полученное время в головную подпрограмму подсистемы BSTPAR9.

В задачу головной подпрограммы BSTPAR9 входит:

— организация циклического обращения к необходимой подпрограмме одного просчета из имеющегося в подсистеме набора (той, которая вызывает для

выполнения расчетов заданную пользователем целевую программу комплекса);

— передача на каждом этапе (шаге) цикла одного из конкретных наборов параметров распараллеливания из множества наборов, заданных пользователем;

— при переходе на новом шаге цикла к решетке процессов с новыми параметрами (NPROW, NPCOL) определения для нее нового контекста и осуществление расчетов с соответствующим новым параметром контекста (CTXT, см. п. 2.4);

— выбор минимального из всех результирующих значений времени, полученных на каждом этапе (шаге) цикла от вызываемой в цикле подпрограммы одного просчета, и выдача этого минимального времени в качестве результата работы подсистемы вместе с соответствующим (наилучшим) набором параметров распараллеливания.

Рассмотрим немного детальнее функции подсистемы BSTPAR. Как уже говорилось в п. 2.5, параметрами распараллеливания называются: NPROW (число строк в решетке процессов), NPCOL (число столбцов в решетке процессов), NB (размер квадратного блока, на которые разбивается исходная матрица).

Пользователь должен указать подсистеме BSTPAR некоторое множество наборов параметров распараллеливания, среди которых она должна выбрать наилучший набор (с минимальным временем счета). Для этого он должен задать пределы изменений указанных параметров.

Так, множество вариантов решеток процессов, описываемых параметрами $\mathrm{NPROW_i}$ и $\mathrm{NPCOL_i}$, где $1 \leqslant \mathrm{i} \leqslant \mathrm{NPROCS}$, должно удовлетворять условию

$$\mathrm{NPROW_i} * \mathrm{NPCOL_i} = \mathrm{NPROCS},$$

где NPROCS — общее число используемых процессов.

Это означает, что в экспериментальных просчетах (одного подхода) могут участвовать только те варианты решеток процессов, которые включают в себя все параллельные процессы, заказанные в команде запуска подсистемы на счет (см. mpirun в п. 7).

При этом пользователь может и должен задать минимальное (NPRWMIN) и максимальное (NPRWMAX) число строк в решетке процессов:

$$1 \leqslant \mathrm{NPRWMIN} \leqslant \mathrm{NPROW_i} \leqslant \mathrm{NPRWMAX} \leqslant \mathrm{NPROCS},$$

в пределах которых будут повторяться расчеты, а также шаг изменения значения $\mathrm{NPROW_i}$ для следующего просчета в цикле (DNPRW).

Соответствующее число столбцов решетки процессов при этом вычисляется по формуле

$$\mathrm{NPCOL_i} = \mathrm{NPROCS/NPROW_i}.$$

Кроме различных вариантов решетки процессов, пользователь может также задать различные варианты третьего параметра распараллеливания NB. За один запуск можно, вообще говоря, перебрать все варианты NB, удовлетворяющие условию

$$2 \leqslant \mathrm{NB} \leqslant \mathrm{N/NPROCS}.$$

Однако для больших матриц не имеет смысла перебирать все NB, удовлетворяющие указанному условию. Пользователю предоставляется возможность за один "заход" произвести расчеты, перебирая варианты NB с некоторым шагом (DNB) от некоторого минимального значения (NBMIN $\geqslant 2$) до некоторого максимального (NBMAX $\leqslant$ N/NPROCS).

Полностью весь набор параметров, передаваемых пользователем в головную подпрограмму BSTPAR9 подсистемы автоматизированного подбора параметров, приводится в описании этой подпрограммы в Приложении 3. Из этого описания видно, что помимо основной функции — выдачи наилучшего набора параметров распараллеливания и соответствующего минимального времени просчета — подпрограмма позволяет с помощью дополнительных параметров управлять такими действиями, как

— выдача на печать исходных данных;

— выдача на печать и/или запись в файл результатов решения задачи с заданной исходной матрицей;

— возможность генерации некоторой "стандартной" исходной матрицы, если пользователь хочет осуществить подбор наилучших параметров без предоставления своего файла с исходной матрицей, и т.п.

4.2. Как уже упоминалось, подсистема BSTPAR включает в себя, помимо программной части, архив результатов.

При любом обращении к этой подсистеме полученный ей результат в виде набора наилучших для конкретного случая параметров распараллеливания запоминается (помещается) в системный архив результатов. Именно это позволяет разным пользователям, желающим проводить вычисления с наилучшей производительностью, не повторять вычисления, связанные с выбором наилучшего набора параметров, если таковой уже имеется для аналогичной задачи в архиве результатов.

Простота устройства архива позволяет пользователю быстро определить, имеется ли в нем интересующий его результат предыдущих обращений к подсистеме BSTPAR, а именно, подходящий набор параметров.

Архив представляет собой обычный древовидный каталог файлов, каждая из ветвей-подкаталогов которого содержит информацию, относящуюся к конкретной целевой программе комплекса PARALG, т.е. к решению задачи линейной алгебры конкретного вида. Имена этих подкаталогов совпадают с именами соответствующих целевых программ. Имена самих файлов, содержащих наилучшие наборы параметров распараллеливания, составлены из чисел, конкретизирующих решавшуюся задачу: порядок исходной матрицы и использовавшееся число параллельных процессов.

Более точно, имя такого системного файла формируется автоматически и строится на основании следующего правила.

Общая структура имени такова:

_N1_N2_D,

где

N1 — порядок исходной матрицы,

N2 — число используемых параллельных процессоров,

D — шестизначное число, обозначающее дату запуска задачи на счет (например, 011214 — означает первое декабря 2014 года).

Начальным символом имени служит "_" (подчерк), такие же символы разделяют и указанные три части имени.

Например, имя файла _7000_8_031014 означает, что он содержит наилучший набор параметров распараллеливания, полученный подсистемой BSTPAR при таймировании результатов решения задачи с исходной матрицей порядка 7000 и при использовании восьми параллельных процессов. При этом таймирование производилось третьего октября 2014 года.

При накапливании в архиве большого количества файлов (т.е. при большом количестве обращений к подсистеме BSTPAR) такая форма имени системных файлов позволяет легко сориентироваться при поисках необходимого результата таймирования, просто посмотрев содержимое подкаталога с именем выбранной целевой программы комплекса.

4.3. Авторами комплекса PARALG проводились различные экспериментальные расчеты с использованием подсистемы BSTPAR на суперкомпьютере "Чебышев" для целевых программ из разных разделов с матрицами разных порядков. Приведем здесь результаты только одного показательного примера.

Проводилось таймирование решения системы линейных уравнений с помощью целевой программы PDGESV с матрицей порядка N = 7000 и при использовании восьми параллельных процессов (NPROCS = 8).

При этом перебирались все возможные для этого случая решетки процессов:

$$\begin{aligned} &\text{NPROW} = 1, \quad &&\text{NPCOL} = 8 \\ &\text{NPROW} = 2, \quad &&\text{NPCOL} = 4 \\ &\text{NPROW} = 4, \quad &&\text{NPCOL} = 2 \\ &\text{NPROW} = 8, \quad &&\text{NPCOL} = 1 \end{aligned}$$

Для каждой из решеток осуществлялся перебор размеров блоков матрицы от NBMIN = 2 до NBMAX = 512 (DNB = 4). Наилучший результат был получен для решетки NPROW = 2 и NPCOL = 4 при значении NB = 14(18). Значение минимального времени счета составило при этом t $\approx$ 4 сек. Наихудший результат был получен при NPROW = 8, NPCOL = 1 и NB = 254. Значение максимального времени счета составило t $\approx$ 7,8 сек.

Это показывает, что при неправильном выборе параметров распараллеливания производительность может ухудшиться почти в 2 раза.

Этот эксперимент показал, что для таких задач предпочтителен выбор решетки, близкой к квадратной, а NB не следует выбирать слишком большим (NB $<$ 32).

Кроме того, построенные графики зависимости времени решения (t) от величины блока матрицы (NB) для всех четырех вариантов решеток процессов показали, что начиная с NB $\approx$ 50 до NB = 254 наблюдается почти линейный рост времени решения t. Затем при NB = 258 во всех четырех случаях происходит резкое падение (уменьшение) времени решения на (0,8 сек; 1,1 сек; 1,35 сек; 1,7 сек). При дальнейшем увеличении NB опять происходит линейное увеличение времени, однако графики более пологие, чем в предыдущем случае.

Аналогичные расчеты с перебором тех же параметров распараллеливания были проделаны и для целевых программ из других функциональных разделов комплекса PARALG. По результатам расчетов были построены графики зависимостей времени решения задач от значений параметров распараллеливания. Гиперссылки на соответствующие файлы с графиками (gf_gesv.xls, gf_posv.xls, gf_syev1.xls, gf_sygv1.xls, gf_geev1.xls, gf_gesvd1.xls) можно найти в системе Интернет по адресу:

http://num-anal.srcc.msu.ru/par_prog/org/bstpar.htm

Для построения графиков использовались возможности программной системы EXCEL.

Ниже представлена сводная таблица полученных результатов для матриц порядка N = 7000 и при использовании 8 параллельных процессов.

Имя п/п-мы	NPROW	NPCOL	NB	t_minim,сек
PDGESV	2	4	14	4.01
PDPOSV	4	2	120	2.58
PDGEEV1	4	2	80	209.0
PDSYEV1	2	4	30	72.4
PDSYGV1	2	4	30	90.3
PDGESVD1	2	4	42	262.0

Проведенные расчеты показывают, что характер графиков различен для разных видов задач. Тем не менее, можно заметить, что для больших размеров матриц (N = 7000) лучшее время достигается при решетке процессов, близкой к квадратной, и при выборе $10 < NB < 120$.

5. Порядок действий и разбор примеров при решении задач линейной алгебры с помощью программ автоматизации.

5.1. Рассмотрим сначала случай, когда пользователь не стремится к "выжиманию" максимальной производительности при параллельном решении задачи линейной алгебры с помощью какой-либо из целевых программ комплекса. Вместо этого он предпочитает (что бывает во многих случаях) сэкономить свои собственные усилия и время при решении вопроса, какие выбрать параметры распараллеливания (см. п. 2.5) и какой объем рабочей памяти необходимо заказать в головной программе для успешного решения задачи.

В этом случае в качестве первого шага ему необходимо обратиться к одной из подпрограмм выбора параметров (см. п. 3.1), которая соответствует его конкретной задаче и решающей ее целевой программе (см. список целевых программ в п. 6).

Рассмотрим конкретный пример действий пользователя и получаемые при этом результаты. Пусть ему требуется решить систему линейных алгебраических уравнений с плотной матрицей общего вида, для чего будет использоваться целевая программа PDGESV. Пусть при этом порядок исходной матрицы системы равен 7000, а число используемых параллельных процессов равно 8. Тогда на первом шаге он должен запустить небольшую обычную последовательную программу, содержащую следующее обращение к подпрограмме предварительного выбора параметров распараллеливания:

```
CALL PAR_GESV (PDGESV, 8, 7000, NPROW, NPCOL, NB, MMIN)
```

После этого он получит следующие значения параметров распараллеливания:

NPROW = 2, NPCOL = 4, NB = 32, MMIN = 8631001

Соответствующий фортранный текст приведен в “Примере использования” описания подпрограммы PAR_GESV в Приложении 1.

5.2. На втором этапе, чтобы самому не формировать правильный список фактических параметров при обращении к целевой программе и не распределять самостоятельно исходную глобальную матрицу системы по параллельным процессам в соответствии с установленными правилами (см. п. 2.3), пользователю необходимо составить головную программу, содержащую обращение к соответствующей подпрограмме вызова (см. п. 3.2, список см. в п. 6). В рассматриваемом примере это подпрограмма CAL_GESV.

CALL CAL_GESV (PDGESV, 7000, 2, 4, AMEM, MEMSIZE, 32, INFILEA, INFILEB, OUTFILE)

Здесь числовые параметры означают порядок исходной матрицы и параметры распараллеливания. Другие параметры имеют следующий смысл:

AMEM — одномерный массив локальной рабочей памяти, где размещаются локальные части исходных глобальных матриц и векторов, промежуточные результаты и локальные части результирующего глобального вектора;

MEMSIZE — размер заказываемой локальной памяти, который должен быть MEMSIZE $\geqslant$ MMIN (см. выше);

INFILEA — имя входного файла, содержащего глобальную исходную матрицу системы A;

INFILEB — имя входного файла, содержащего глобальный исходный вектор правой части системы B;

OUTFILE — имя файла, содержащего выходной вектор результатов решения системы.

Фортранный текст соответствующей головной программы а также результаты решения системы приведены в “Примере использования” описания подпрограммы CAL_GESV в Приложении 2.

Подпрограмма вызова CAL_GESV содержит обращение к подпрограмме PDLAREAD, которая занимается чтением из файлов и распределением по параллельным процессам исходных матриц и векторов. При этом предъявляются следующие требования к структуре входных файлов: вначале они должны со-

держать два целых числа, соответствующих размерам (M, N) матриц или векторов. В рассматриваемом примере это: 7000 7000 (для файла INFILEA), и 7000 1 (для файла INFILEB). Далее элементы матрицы должны быть расположены в один столбец друг за другом по столбцам (как принято в языке ФОРТРАН).

Предполагается, что каждая матрица или вектор хранится в отдельном файле указанной структуры.

В случае каких-либо отличий в структуре входных файлов, необходимо или перезаписать их в соответствии с указанными требованиями, или внести необходимые правки в подпрограмму PDLAREAD.

По окончании выполнения второго шага, полученный вектор результатов решения системы будет записан (в соответствии с этими же правилами) в выходной файл OUTFILE, а также выдан на печать.

5.3. Рассмотрим теперь случай ситуации, когда пользователь хочет подобрать для решения своей конкретной задачи наилучшие (с точки зрения производительности) параметры распараллеливания. При этом решать необходимо ту же самую конкретную задачу, которая была рассмотрена в п. 5.1.

Предположим также, что в описанном в п. 4.2 системном архиве подсистемы BSTPAR еще нет наилучшего набора параметров распараллеливания для указанного конкретного случая (целевая программа — PDGESV, порядок матрицы равен 7000, число используемых процессов равно 8). Иными словами, никто из пользователей данной вычислительной установки (данного суперкомпьютера) еще не обращался к подсистеме BSTPAR с указанными параметрами конкретной задачи.

В этом случае пользователю необходимо составить головную программу с обращением к головной подпрограмме BSTPAR9 подсистемы подбора наилучших параметров распараллеливания. Для рассматриваемого случая обращение может выглядеть так:

```
CALL BSTPAR9 ('PDGESV', 7000, 7000, 8, AMEM, 8650000, 2, 250, 4, 1, 8, 1,
                INFILEA, INFILEB, OUTFILE, TFILE, 2)
```

Это означает, что пользователь пожелал протаймировать все случаи решения задачи со всеми вариантами решеток процессов с количеством строк $1 \leqslant \mathrm{NPROW} \leqslant 8$ и всеми вариантами блоков разбиения матриц размеров $2 \leqslant \mathrm{NB} \leqslant 250$ с шагом 4 (т.е. $2, 6, 10, \ldots, 250$).

Пример полного фортранного варианта такой головной программы приведен в Приложении 3 в “Примере использования”.

Полученный таким образом наилучший набор параметров распараллеливания (NPROW = 2, NPCOL = 4, NB = 18, MMIN = 8631001) можно затем неоднократно использовать в качестве входных параметров при обращении к

подпрограмме CAL_GESV (см. п. 5.2) для решения систем с разными матрицами того же порядка (7000).

6. Общий список надстроечных подпрограмм комплекса PARALG.

6.1. Список подпрограмм комплекса PARALG, предназначенных для выбора параметров распараллеливания целевых программ:

— решение систем линейных алгебраических уравнений

PAR_GESV	с вещественной матрицей общего вида методом Гаусса
PAZ_GESV	с комплексной матрицей общего вида методом Гаусса
PAR_POSV	с симметричной положительно определенной матрицей методом квадратного корня (методом Холецкого)
PAZ_POSV	с эрмитовой положительно определенной матрицей методом квадратного корня (методом Холецкого)
PAR_BDSV	с вещественной ленточной матрицей
PAZ_BDSV	с ленточной комплексной матрицей
PAR_TRSV	с вещественной трехдиагональной матрицей
PAZ_TRSV	с трехдиагональной комплексной матрицей

— вычисление собственных значений матриц в линейной постановке

PAR_GEEV1	для вещественной матрицы общего вида
PAR_SYEV1	для вещественной симметричной матрицы
PAZ_HEEV1	для эрмитовой матрицы

— вычисление собственных значений и собственных векторов матриц в линейной постановке

PAR_SYEVZ	для вещественной симметричной матрицы
PAZ_HEEVZ	для эрмитовой матрицы

— вычисление собственных значений матриц в обобщенной постановке

PAR_SYGV1	для вещественных симметричных матриц
PAZ_HEGV1	для эрмитовых матриц

— вычисление собственных значений и собственных векторов матриц в обобщенной постановке

PAR_SYGVZ	для вещественных симметричных матриц
PAZ_HEGVZ	для эрмитовых матриц

— вычисление сингулярных чисел и сингулярных векторов матриц

PAR_GESVD1 сингулярных чисел матриц
PAR_GESVD сингулярных чисел и сингулярных векторов матриц

6.2. Список подпрограмм комплекса PARALG, предназначенных для автоматизации вызова целевых программ:

— решение систем линейных алгебраических уравнений

CAL_GESV с вещественной матрицей общего вида методом Гаусса
CAZ_GESV с комплексной матрицей общего вида методом Гаусса
CAL_POSV с симметричной положительно определенной матрицей методом квадратного корня (методом Холецкого)
CAZ_POSV с эрмитовой положительно определенной матрицей методом квадратного корня (методом Холецкого)
CAL_BDSV с вещественной ленточной матрицей
CAZ_BDSV с ленточной комплексной матрицей
CAL_TRSV с вещественной трехдиагональной матрицей
CAZ_TRSV с трехдиагональной комплексной матрицей

— вычисление собственных значений матриц в линейной постановке

CAL_GEEV1 для вещественной матрицы общего вида
CAL_SYEV1 для вещественной симметричной матрицы
CAZ_HEEV1 для эрмитовой матрицы

— вычисление собственных значений и собственных векторов матриц в линейной постановке

CAL_SYEVZ для вещественной симметричной матрицы
CAZ_HEEVZ для эрмитовой матрицы

— вычисление собственных значений матриц в обобщенной постановке

CAL_SYGV1 для вещественных симметричных матриц
CAZ_HEGV1 для эрмитовых матриц

— вычисление собственных значений и собственных векторов матриц в обобщенной постановке

CAL_SYGVZ для вещественных симметричных матриц
CAZ_HEGVZ для эрмитовых матриц

— вычисление сингулярных чисел и сингулярных векторов матриц

CAL_GESVD1 сингулярных чисел матриц
CAL_GESVD сингулярных чисел и сингулярных векторов матриц

6.3. Подпрограммы подсистемы BSTPAR автоматизированного подбора наилучших параметров распараллеливания:

— головная подпрограмма

BSTPAR9 подпрограмма организации цикла по вызову целевых программ с разными фактическими параметрами распараллеливания и выдачи наилучшего набора таких параметров

— набор подпрограмм одного просчета, вызываемых из головной подпрограммы BSTPAR9 и организующих вызов одной из целевых подпрограмм комплекса и фиксирующих время просчета:

CAL_GESVN для решения систем линейных алгебраических уравнений с невырожденными матрицами общего вида
CAL_POSVN для решения систем линейных алгебраических уравнений с симметричными матрицами
CAL_SYEV1N для решения линейной проблемы собственных значений с симметричной матрицей
CAL_SYGV1N для решения обобщенной проблемы собственных значений с симметричной матрицей
CAL_GEEV1N для решения линейной проблемы собственных значений с матрицей общего вида
CAL_GESVD1N для вычисления сингулярных чисел матрицы

Полные описания всех указанных в п. 6 подпрограмм автоматизации приведены в “Систематическом каталоге” Руководства по использованию комплекса PARALG

http://num-anal.srcc.msu.ru/par-prog/comparp.htm

7. Запуск программ комплекса в ОС Linux на суперкомпьютере “Чебышёв” при использовании средств автоматизации.

Комплекс программ PARLAG доступен пользователям суперкомпьютера “Чебышёв” в НИВЦ МГУ в виде откомпилированных библиотек. Пакеты BLAS, BLACS, ScaLAPACK и др., модули которых используются программами комплекса, входят в состав MKL-библиотеки (Intel Math Kernel Library), установленной на этом суперкомпьютере, и должны быть подсоединены при получении исполняемой программы. Объектные модули самого комплекса берутся из

библиотеки libparalg.a, указанной в последней строке приводимого ниже скрипта bld-tcl. Здесь представлен общий вид этого скрипта, предназначенного для компиляции и получения исполняемой программы, вызывающей подпрограммы комплекса PARALG, которая может быть затем запущена на счет в параллельном режиме на суперкомпьютере “Чебышёв”.

```
#! /bin/sh
EXEDIR=.
EXE="<имя исполнимой программы>"
OBJ="<список имен объектных модулей>"
F77OPTS="-O2"
F77=mpif77
#
$F77 -c $F77OPTS <имя исходного фортранного модуля>
#
echo Linking executable $EXE
#
$F77 -o $EXEDIR/$EXE $OBJ \
       -Wl,--start-group -lmkl_scalapack_lp64 -lmkl_blacs_lp64 \
       -lmkl_intel_lp64 \
       -lmkl_sequential -lmkl_core \
       -Wl,--end-group \
       -lpthread -libparalg.a
```

Здесь:

- <имя исполняемой программы> — указываемое пользователем имя, которое будет присвоено полученному исполняемому модулю (например, tpdgesv.e);
- <список имен объектных модулей> — список имен всех объектных модулей, которые не вызываются из библиотек (например, tpdgesv.o);
- $F77 -c $F77OPTS <имя исходного фортранного модуля> — команда компиляции (которых может быть несколько) фортранного модуля (например, $F77 -c $F77OPTS tpdgesv.f).

В скрипте bld-tcl подключаются все пакеты из MKL-библиотеки, содержащие все вызываемые подпрограммы, используемые при работе целевых параллельных программ комплекса.

Полученная исполняемая программа может быть затем запущена на счет с помощью команды mpirun с параметрами.

Например, если требуется запустить программу tpdgesv.e на шести процессорах, то команда запуска имеет вид

mpirun -np 6 -maxtime 1 ./tpdgesv.e

В тестах к программам выдача исходных матриц и полученных результатов осуществляется с помощью обращения к подпрограмме PDLAPRNT из пакета ScaLAPACK(TOOLS). Эта подпрограмма распечатывает всю распределенную матрицу (вектор) полностью по столбцам по одному элементу в строке, получая элементы со всех участвующих в вычислениях процессоров.

Приведем далее конкретный вид скрипта для случая решения задач с использованием описанных в настоящем пособии средств автоматизации (см. также п.п. 3, 5).

Так, если система линейных уравнений решается с помощью целевой программы комплекса PDGESV с использованием подпрограммы автоматизации вызова CAL_GESV.f, описанной в п.п. 3.2, 5.2 (см. также Приложение 2), то текст скрипта bld-tcl будет иметь вид

```
#! /bin/sh
EXEDIR=.
EXE="tcl_gesv.e"
OBJ="tcl_gesv.o cal_gesv.o pdgesv.o pdlaprnt.o pdlaread.o pdlawrite.o"
#
F77OPTS="-O2"
F77=mpif77
#
$F77 -c $F77OPTS tcl_gesv.f
#
$F77 -o $EXEDIR/$EXE $OBJ \
       -Wl,--start-group -lmkl_scalapack_lp64 -lmkl_blacs_lp64 \
       -lmkl_intel_lp64 \
       -lmkl_sequential -lmkl_core \
       -Wl,--end-group \
       -lpthread -libparalg.a
```

Чтобы не указывать все объектные модули подпрограмм комплекса PARALG, которые необходимо вызвать для решения конкретной задачи, достаточно указать в последней строке скрипта объектную библиотеку libparalg.a. Тогда строка с объектными модулями будет содержать только объектный модуль головной программы OBJ=“tcl_gesv.o”.

Здесь подпрограммы pdlaread.o и pdlawrite.o — входящие в состав комплекса служебные подпрограммы для считывания из файла исходных матриц и распределения их по параллельным процессам и записи в файл полученного вектора решения.

При желании пользователя подобрать наилучшие параметры распараллеливания при решении рассмотренной в п. 5.3 задачи, исполняемая программа может быть получена с помощью аналогичного скрипта, в котором необходимо заменить три строки на указанные ниже:

```
EXE="tbstpar.e"

OBJ="tbstpar.o bstpar9.o cal_gesvn.o pdgesv.o pdlaprnt.o pdlaread.o
    pdlawrite.o"

$F77 -c $F77OPTS tbstpar.f
```

Команда запуска на счет этой программы имеет вид

mpirun -np 8 -maxtime 3000 ./tbstpar.e

Большое астрономическое время (в минутах), заказанное в этой команде, требуется из-за очень большого числа перебираемых вариантов наборов параметров распараллеливания, указанных в примере в п. 5.3 (см. также “Пример использования” в Приложении 3).

Приложение 1

Пример описания подпрограммы автоматизации выбора параметров распараллеливания для целевых программ.

Подпрограмма: PAR_GESV

Назначение

Выбор параметров распараллеливания для целевых программ комплекса PARALG, решающих системы линейных алгебраических уравнений на распределенной памяти с матрицами общего вида

Описание

Эта служебная подпрограмма выбирает и вычисляет за пользователя параметры распараллеливания, которые конкретизируют распределение вычислений между параллельными процессами и значения которых необходимо передавать целевым программам решения систем линейных алгебраических уравнений с матрицами общего вида

Литература:

1. http://num_anal.srcc.msu.ru/par_prog/cat562n.htm
2. http://software.intel.com/en-us/articles/intel-math-kernel-library-documentation/
3. http://www.netlib.org/scalapack/slug/index.html

Использование

CALL PAR_GESV(NAME, N, NPROCS, MMIN, NB, NPROW, NPCOL)

Параметры

NAME — имя целевой программы, с помощью которой пользователь собирается решать свою задачу (одно из PDGESV, PDGESV1, PDGESV2, PDGESV3); (входной параметр, тип символьный);

N — порядок исходной матрицы системы (входной параметр, тип целый);

NPROCS — число процессов, которое может выделить пользователь для решения задачи (входной параметр, тип целый);

MMIN — минимальное количество локальной памяти, которое необходимо выделить на каждом из параллельных процессов для решения задачи (выходной параметр, тип целый);

NB — величина блока, на которые делится исходная матрица системы; NB полагается равным 0, если пользователь доверяет выбор его значения этой подпрограмме (входной и выходной параметр, тип целый);

NPROW — число строк в решетке процессов (выходной параметр, тип целый);

NPCOL — число столбцов в решетке процессов (выходной параметр, тип целый);

Вызываемые подпрограммы: нет

Замечания по использованию

1. Подпрограмма является последовательной фортранной программой и не использует распределенную память (требуется только один процессор).

2. Используются подпрограммы ICEIL, ILCM, NUMROC (из библиотеки ScaLAPACK_TOOLS).

Пример использования

Необходимо выбрать размер блока NB, число строк и столбцов решетки процессов (NPROW, NPCOL) и вычислить размер локальной памяти, необходимой для решения системы уравнений с помощью подпрограммы PDGESV.
Матрица системы — квадратная общего вида порядка 9.
Выделяемое число процессов NPROCS = 4. NB = 0.

Фрагмент фортранного текста вызывающей программы

Полный текст теста можно получить в tpr_gesv.zip.

```
      PROGRAM TPR_GESV
*
      INTEGER             N, NPROCS, MMIN, NB, NPROW, NPCOL
      CHARACTER*(*)       NAME
      PARAMETER           ( NAME = 'PDGESV')
      EXTERNAL PAR_GESV
*
      N = 9
      NPROCS = 4
      NB=0
*
      CALL  PAR_GESV(NAME, N, NPROCS, MMIN, NB, NPROW, NPCOL)
      STOP
      END
```

Результаты:

```
NB = 4
NPROW = 2
NPCOL = 2
MMIN = 41
```

Приложение 2

Пример описания подпрограммы автоматизации вызова целевой программы.

Подпрограмма: CAL_GESV

Назначение

Автоматизация подготовки входных данных и вызова целевых программ решения систем линейных алгебраических уравнений с невырожденными матрицами общего вида.

Описание

Эта служебная подпрограмма организует за пользователя правильный вызов целевых программ решения систем линейных алгебраических уравнений с невырожденными матрицами общего вида. Она считывает из внешних файлов исходную матрицу системы и вектор (матрицу) правой части, распределяет их части (блоки) по параллельным процессам в соответствии с используемыми алгоритмами и формирует правильный список фактических параметров для выбранной пользователем целевой программы. После чего осуществляет вызов этой программы. Полученный в результате счета вектор решения задачи записывается во внешний файл.

Литература:
1. http://num_anal.srcc.msu.ru/par_prog/cat562n.htm
2. http://software.intel.com/en-us/articles/intel-math-kernel-library-documentation/
3. http://www.netlib.org/scalapack/slug/index.html

Использование

CALL CAL_GESV(NAME, TRANS, N, NPROW, NPCOL, AMEM, MEMSIZE, NB, INFILEA, INFILEB, OUTFILE)

Параметры

NAME — имя целевой подпрограммы, с помощью которой пользователь собирается решать свою задачу (одно из PDGESV, PDGESV1, PDGESV2, PDGESV3); (входной параметр, тип символьный);

TRANS — переменная символьного типа, означающая, следует ли решать систему с транспонированной матрицей и принимающая значения “N” или “T” (входной параметр);

N — порядок исходной матрицы системы (входной параметр, тип целый);

NPROW — число строк в решетке процессов, которое было выдано пользователю соответствующей подпрограммой вычисления параметров PAR_GESV (см. п. 5.1 и Приложение 1) (входной параметр, тип целый);

NPCOL — число столбцов в решетке процессов, которое было выдано пользователю соответствующей подпрограммой вычисления параметров PAR_GESV (см. п. 5.1 и Приложение 1) (входной параметр, тип целый);

AMEM — массив локальной рабочей памяти, в которую распределяются локальные части исходных матриц и векторов систем уравнений, хранятся элементы рабочих массивов, где формируются локальные части вектора решений системы (входной параметр, тип DOUBLE PRECISION);

MEMSIZE — задаваемый размер массива AMEM, который должен быть больше или равен значению параметра MMIN, выданному пользователю соответствующей подпрограммой вычисления параметров PAR_GESV (см. п. 5.1 и Приложение 1) (входной параметр, тип целый);

NB — размер блока, матрицы, выданный пользователю соответствующей подпрограммой вычисления параметров PAR_GESV (или фигурировавший в качестве входного значения при вычислении значений других параметров) (см. п. 2.5, п. 5.1 и Приложение 1) (входной параметр, тип целый);

INFILEA — имя файла, содержащего исходную матрицу системы (A) (входной параметр, тип символьный);

INFILEB — имя файла, содержащего исходный вектор правой части системы (B) (входной параметр, тип символьный);

OUTFILE — имя файла, в который в результате работы подпрограммы будет записан результирующий вектор (X) (входной параметр, тип символьный).

Вызываемые подпрограммы

Здесь указаны только целевые (1-ого уровня) и базовые подпрограммы (2-ого уровня), которые вызываются из целевых подпрограмм.

PDGESV — решение системы $AX = B$ с матрицей общего вида методом
PDGESV1 Гаусса с выбором ведущего элемента по столбцу для
PDGESV2 вещественных данных двойной точности или

PDGESV3		решение системы AX = B или $A^TX = B$ с матрицей общего вида методом Гаусса с выбором ведущего элемента по столбцу для вещественных данных двойной точности или решение системы AX = B или $A^TX = B$ с матрицей общего вида методом Гаусса с выбором ведущего элемента по столбцу для вещественных данных двойной точности и оценка обратного числа обусловленности, или решение системы AX = B или $A^TX = B$ с матрицей общего вида методом Гаусса с выбором ведущего элемента по столбцу для вещественных данных двойной точности с итерационным уточнением решения и оценкой границ ошибок соответственно
PDGETRF PZGETRF	—	LU — разложение матрицы общего вида методом Гаусса с выбором ведущего элемента по столбцу
PDGETRS PZGETRS	—	Решение системы AX = B, $A^TX = B$ или $A^HX = B$ на основе LU — разложения, полученного подпрограммой PDGETRF(PZGETRF)
PDGECON PZGECON	—	Оценка обратного числа обусловленности матрицы общего вида
PDGERFS PZGERFS	—	Выполнение итерационного уточнения решения системы AX = B, $A^TX = B$ или $A^HX = B$, полученного подпрограммой PDGETRS(PZGETRS), и оценка границ ошибок полученного решения

Замечания по использованию

Используются следующие подпрограммы из пакета BLACS в составе MKL-библиотеки: BLACS_EXIT, BLACS_GET, BLACS_GRIDEXIT, BLACS_GRIDINFO, BLACS_GRIDINIT, BLACS_PINFO, BLACS_SETUP

Используются подпрограммы DESCINIT, ICEIL, NUMROC, PDLAPRNT из пакета ScaLAPACK(TOOLS) в составе MKL-библиотеки.

Кроме того, используются подпрограммы PDLAREAD и PDLAWRITE.

Пример использования

Необходимо решить систему уравнений с помощью подпрограммы PDGESV.
Матрица системы — общего вида порядка 9.
Пусть матрица A системы имеет вид:

$$\begin{pmatrix} 19 & 3 & 1 & 12 & 1 & 16 & 1 & 3 & 11 \\ -19 & 3 & 1 & 12 & 1 & 16 & 1 & 3 & 11 \\ -19 & -3 & 1 & 12 & 1 & 16 & 1 & 3 & 11 \\ -19 & -3 & -1 & 12 & 1 & 16 & 1 & 3 & 11 \\ -19 & -3 & -1 & -12 & 1 & 16 & 1 & 3 & 11 \\ -19 & -3 & -1 & -12 & -1 & 16 & 1 & 3 & 11 \\ -19 & -3 & -1 & -12 & -1 & -16 & 1 & 3 & 11 \\ -19 & 3 & -1 & -12 & -1 & -16 & -1 & 3 & 11 \\ -19 & -3 & -1 & -12 & -1 & -16 & -1 & -3 & 11 \end{pmatrix}$$

Вектор правых частей B имеет вид:

$$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

При предварительном обращении к подпрограмме PAR_GESV были получены следующие значения необходимых параметров.
Решетка процессов: NPROW = 2, NPCOL = 2; NB = 4 (т.е. матрица разбивается на блоки размера 4 на 4), MMIN = 41.

Фрагмент фортранного текста вызывающей программы

Полный текст теста можно получить в tcl_gesv.zip.

```
      PROGRAM TCL_GESV
      include 'mpif.h'
      INTEGER                 N, NPROW, NPCOL, NB, MEMSIZE
      PARAMETER           ( N=9, NB=4, MEMSIZE=41)
*
      DOUBLE PRECISION    AMEM( MEMSIZE )
      CHARACTER*(*)            NAME, TRANS
      CHARACTER           INFILEA*(*), INFILEB*(*), OUTFILE*(*)
      PARAMETER          ( INFILEA = 'gesvA_9.dat',
     $                               INFILEB = 'gesvB_9.dat',
```

```
     $                                  OUTFILE = 'gesv_9.res')
*
      PARAMETER          ( TRANS = 'N',
     $                                  NAME = 'PDGESV')
      EXTERNAL CAL_GESV
*
      NPROW = 2
      NPCOL = 2
*
      CALL CAL_GESV(NAME, TRANS, N, NPROW, NPCOL, AMEM, MEMSIZE, NB,
     $                                  INFILEA, INFILEB, OUTFILE)
*
      STOP
      END
```

Результаты:

Решение системы (в файле OUTFILE)

```
X = ( 0.D0, - 0.16666666666667D0, 0.5D0, 0.D0, 0.D0, 0.D0, - 0.5D0,
    0.16666666666667D0, 0.D0 )
```

Приложение 3

Головная подпрограмма подсистемы BSTPAR.

Подпрограмма: BSTPAR9

Назначение

Головная подпрограмма подсистемы подбора наилучших (с точки зрения производительности) параметров распараллеливания BSTPAR. Организует циклический вызов какой-либо целевой программы комплекса PARALG решения задач линейной алгебры на распределенной памяти с передачей на каждом шаге цикла разных наборов параметров распараллеливания. В результате выдает наилучший набор таких параметров.

Описание

Эта подпрограмма организует циклическое обращение к заданной пользователем целевой подпрограмме комплекса PARALG, решающей одну из задач линейной алгебры с разными наборами фактических параметров распараллеливания (NPROW, NPCOL, NB) из заданного пользователем диапазона их допустимых значений. По результатам таймирования, производимого на каждом шаге цикла, она выбирает и выдает наилучший набор параметров распараллеливания, соответствующий наименьшему времени, затраченному на решение данной задачи линейной алгебры.

Литература:
1. http://software.intel.com/en-us/articles/intel-math-kernel-library-documentation/
2. http://www.netlib.org/scalapack/slug/index.html
3. http://num_anal.srcc.msu.ru/par_prog/

Использование

CALL BSTPAR9 (NAME, M, N, NPROCS, AMEM, MEMSIZE, NBMIN, NBMAX, DNB, NPRWMIN, NPRWMAX, DNPRW, INFILEA, INFILEB, OUTFILE, TFILE, NPRT)

Параметры

NAME — имя целевой подпрограммы, при применении которой для решения своей задачи пользователь хочет подобрать наилучшие параметры распараллеливания (одно из: PDGESV, PDGESV1, PDGESV2, PDGESV3, PDPOSV, PDPOSV1, PDPOSV2, PDGEEV1, PDSYEV1, PDSYEV3, PDSYEV5, PDSYGV1, PDSYGV3, PDSYGV5, PDGESVD1, PDGESVD4); (входной параметр, тип символьный);

M — число строк исходной матрицы, для которого необходимо подобрать наилучшие параметры распараллеливания (входной параметр, тип целый);

N — число столбцов исходной матрицы, для которого необходимо подобрать наилучшие параметры распараллеливания (входной параметр, тип целый);

NPROCS — число параллельных процессов, выделенных для решения задачи (входной параметр, тип целый);

AMEM — массив локальной рабочей памяти, в которую распределяются локальные части исходных матриц и векторов, хранятся элементы рабочих массивов, где формируется массив решений; (входной параметр, тип DOUBLE PRECISION);

MEMSIZE — задаваемый размер массива AMEM, который должен быть больше или равен наибольшему значению параметра MMIN, выданному пользователю соответствующей подпрограммой вычисления параметров (например, PAR_GESV — при решении системы уравнений) при задании ей различных наборов параметров распараллеливания, для которых проводится таймирование (см. Автоматизация доступа к подпрограммам Комплекса, п. 3) (входной параметр, тип целый);

NBMIN — минимальный размер блока матрицы (NB) из заданного пользователем диапазона значений, при которых будет осуществляться решение задачи и подсчитываться время, затраченное на это решение (входной параметр, тип целый);

NBMAX — максимальный размер блока матрицы (NB) из заданного пользователем диапазона значений, при которых будет осуществляться решение задачи и подсчитываться время, затраченное на это решение (входной параметр, тип целый);

DNB — размер шага изменения величины блока для вычисления очередного значения NB (из заданного пользователем диапазона), при котором будет выполняться решение задачи ($1 \leqslant \mathrm{DNB} \leqslant \mathrm{NBMAX}$) и подсчитываться время, затраченное на это решение (входной параметр, тип целый);

NPRWMIN — минимальное число строк в решетке процессов, которая будет использоваться при решении задачи, из заданного пользователем диапазона возможных решеток, при которых будут выполняться расчеты и подсчитываться затраченное время (входной параметр, тип целый);

NPRWMAX — максимальное число строк в решетке процессов, которая будет использоваться при решении задачи, из заданного пользователем диапазона возможных решеток, при которых будут выполняться расчеты и подсчитываться затраченное время (входной параметр, тип целый);

DNPRW — размер шага изменения числа строк в решетке процессов (NPROW) для вычисления очередного значения NPROW (из заданного пользователем диапазона), при котором будет выполняться решение задачи ($1 \leqslant \mathrm{DNPRW} \leqslant \mathrm{NPRWMAX}$) и подсчитываться время, затраченное на это решение (входной параметр, тип целый);

INFILEA — имя файла, содержащего исходную матрицу (A), либо “ ” (пробел); в последнем случае будет использована некая “стандартная” матрица заданного порядка (входной параметр, тип символьный);

INFILEB — при подборе параметров для задачи решения системы линейных уравнений — имя файла, содержащего исходный вектор правой части системы (B), либо “ ” (пробел); в последнем случае будет использован некий “стандартный” вектор заданного порядка;
при подборе параметров для задачи решения линейной проблемы собственных значений или задачи вычисления сингулярных чисел — не используется;
при подборе параметров для задачи решения обобщенной проблемы собственных значений — имя файла, содержащего исходную матрицу (B), либо “ ” (пробел); в последнем случае будет использована некая “стандартная” матрица заданного порядка; (входной параметр, тип символьный);

OUTFILE — имя файла, в который (если это необходимо, см. параметр NPRT) будут записаны результаты работы целевой вычислительной подпрограммы с именем NAME (входной параметр, тип символьный);

TFILE — имя файла, в который в результате нескольких расчетов по целевой подпрограмме будут записаны все результаты таймирования, полученные при всех вычислительных экспериментах (т.е. на всех шагах цикла), выполненных в данном запуске подпрограммы BSTPAR9, инициирующей групповой запуск нескольких вычислительных экспериментов (т.е. нескольких вызовов одной из подпрограмм: CAL_GESVN, CAL_POSVN, CAL_GEEV1N, CAL_SYEV1N, CAL_SYGV1N или CAL_GESVD1N, с разными наборами параметров распараллеливания) (входной параметр, тип символьный);

NPRT — режим выдачи результатов вычислительного эксперимента:

= 0 — на печать не выдается ничего, кроме минимального времени счета полученного в данном групповом эксперименте и значений наилучших параметров распараллеливания, при которых получено это минимальное время;

= 1 — на печать выдаются все результаты таймирования (при всех испытанных в данном групповом запуске наборах параметров распараллеливания);

= 2 — на печать выдается все то же самое, что и при NPRT = 1, а также результаты решения самой задачи целевой программой комплекса для заданной матрицы;

= 3 — на печать выдается все то же самое, что и при NPRT = 2, а также все входные параметры и матрицы (входной параметр, тип целый);

Вызываемые подпрограммы

Здесь указаны только подпрограммы 1-ого и 2-ого уровня.
Для того чтобы упростить вызов всех используемых в решении задачи подпрограмм Комплекса достаточно при получении исполнимого модуля подсоединить объектную библиотеку libparalg.a (см., например, п. 7 данного учебного пособия).

CAL_GESVN — Организация вызова целевой подпрограммы комплекса для решения систем линейных алгебраических уравнений с невырожденными матрицами общего вида (одной из: PDGESV, PDGESV1, PDGESV2, PDGESV3) и вычисление времени, затраченного на решение задачи

CAL_POSVN — Организация вызова целевой подпрограммы комплекса для решения систем линейных алгебраических уравнений с симметричными матрицами (одной из: PDPOSV, PDPOSV1, PDPOSV2) и вычисление времени, затраченного на решение задачи

CAL_SYEV1N — Организация вызова целевой подпрограммы комплекса для решения линейной проблемы собственных значений с симметричной матрицей (одной из: PDSYEV1, PDSYEV3, PDSYEV5) и вычисление времени, затраченного на решение задачи

CAL_SYGV1N — Организация вызова целевой подпрограммы комплекса для решения обобщенной проблемы собственных значений с симметричной матрицей (одной из: PDSYGV1, PDSYGV3, PDSYGV5) и вычисление времени, затраченного на решение задачи

CAL_GEEV1N — Организация вызова целевой подпрограммы комплекса для решения линейной проблемы собственных значений с матрицей общего вида (PDGEEV1) и вычисление времени, затраченного на решение задачи

CAL_GESVD1N — Организация вызова целевой подпрограммы комплекса для вычисления сингулярных чисел матрицы (одной из: PDGESVD1, PDGESVD4) и вычисление времени, затраченного на решение задачи

MINTGESV — Построение некоторых "стандартных" исходных матриц (и
MINTPOSV векторов) заданного порядка N и распределение их час-
MINTSYEV1 тей (блоков) по параллельным процессам, если делается
MINTSYGV1 подбор параметров для задачи:
MINTGEEV1 - решения системы линейных уравнений с матрицами об-
MINTGESVD1 щего вида,
MINTGESVD2 - решения системы линейных уравнений с симметричны-
ми матрицами,

- вычисления собственных значений симметричных матриц в линейной проблеме собственных значений,
- вычисления собственных значений симметричных матриц в обобщенной проблеме собственных значений,
- вычисления собственных значений матриц общего вида в линейной проблеме собственных значений,
- вычисления сингулярных чисел квадратной матрицы,

- вычисления сингулярных чисел прямоугольной матрицы, соответственно;

в том случае, если не были заданы файлы с исходными матрицами или векторами.

Замечания по использованию

1. Используются следующие подпрограммы из пакета BLACS в составе MKL-библиотеки:

BLACS_EXIT, BLACS_GET, BLACS_GRIDEXIT,BLACS_GRIDINFO, BLACS_GRIDINIT, BLACS_PINFO, BLACS_SETUP, BLACS_PNUM.

Используется подпрограмма SL_GRIDRESHAPE из пакета ScaLAPACK(TOOLS) в составе MKL-библиотеки. Кроме того, используются подпрограммы PDLAREAD, PDLAWRITE и MPI_Wtime( ).

2. Минимальное время, затраченное на решение задачи, среди всех времен, полученных при всех расчетах с разными параметрами распараллеливания, использованными в данном групповом запуске, присваивается переменной **time**, описанной в общем блоке.

3. Для сохранения, систематизации и обеспечения общего доступа ко всем, полученным на данном компьютере, результатам вычисления минимального времени, необходимого для расчетов по той или иной целевой программе комплекса PARALG при заданных: порядке матрицы и числе использованных процессоров, создается некий системный архив файлов с результатами всех запусков BSTPAR9.

Каждый такой файл содержит минимальное время решения задачи и искомые параметры распараллеливания, при которых оно было достигнуто. Имена таких файлов строятся системно и состоят из указания порядка матрицы, числа процессоров и даты запуска программы BSTPAR9. Это облегчает поиск нужного файла с лучшими параметрами распараллеливания, позволяя разным пользователям в ряде случаев воспользоваться уже полученными кем-то наилучшими параметрами распараллеливания, избежав собственных значительных затрат времени на аналогичные расчеты.

4. Для того чтобы упростить вызов всех используемых в решении задачи подпрограмм комплекса, достаточно при получении исполнимого модуля подсоединить объектную библиотеку libparalg.a (см., например, п. 7 данного учебного пособия).

Пример использования

Требуется подобрать параметры распараллеливания, наилучшие с точки зрения производительности, при решении системы линейных уравнений с помощью подпрограммы PDGESV.

Матрица системы — порядка 7000. Для решения задачи предполагается использовать 8 процессоров.
При этом предполагается провести вычисления для всех возможных решеток, в которые можно организовать все 8 процессоров (1×8, 2×4, 4×2, 8×1), т.е. параметры решетки изменяются в диапазоне NPRWMIN = 1, NPRWMAX = 8, DNPRW = 1.
Кроме того, для каждой из указанных решеток предполагается провести несколько разных вычислений при разбиении исходной матрицы на блоки разной величины, которая задается диапазоном: NBMIN = 2, NBMAX = 512, DNB = 4.
При предварительном обращении к подпрограмме PAR_GESV с указанными выше параметрами был вычислен максимальный размер локальной памяти (для разных наборов параметров распараллеливания), необходимой для решения задачи MMIN = 8631001.
При вычислениях использовалась "стандартная" матрица, построение и распределение которой по процессам выполняется служебной подпрограммой MINTGESV.

Фрагмент фортранного текста вызывающей программы

```
      PROGRAM  TBSTPAR
*
*     Тест к подпрограмме подбора параметров BSTPAR9
*
      include 'mpif.h'
*
      CHARACTER*(*)  NAME
      PARAMETER      ( NAME = 'PDGESV' )
      CHARACTER      INFILEA*(*), INFILEB*(*), OUTFILE*(*), TFILE*(*)
      PARAMETER      ( INFILEA = ' ', INFILEB = ' ',
     $                 OUTFILE = 'BSTRES.RES', TFILE = 'BSTPART.RES' )
*
      INTEGER        INFO, MYCOL, MYROW, NPCOL, NPROW, DNB, DNPRW,
     $               IAM, MEMSIZE, NPRC, NPRT, NPRMAX, NPRMIN,
     $               M, N, NBMIN, NBMAX, NPROCS
*
      PARAMETER      ( N = 7000, MEMSIZE = 8650000, M = N,
     $                 NPRC = 8,
     $                 NPRWMIN = 1, NPRWMAX = 8 , DNPRW = 1,
     $                 NBMIN = 2, NBMAX = 512, DNB= 4, NPRT = 2 )
*
      COMMON                   time
```

```
*
      DOUBLE PRECISION    AMEM( MEMSIZE ), time
*
      EXTERNAL           BSTPAR9, BLACS_PINFO
*
      CALL  BLACS_PINFO( IAM, NPROCS )
*
      CALL  BSTPAR9( NAME, M, N, NPROCS, AMEM, MEMSIZE, NBMIN, NBMAX,
     $               DNB, NPRWMIN, NPRWMAX, DNPRW, INFILEA,
     $               INFILEB, OUTFILE, TFILE, NPRT )
*
      STOP
      END
```

Результаты:
Минимальное время решения задачи time = 0.402D+01
получено при следующих параметрах распараллеливания:
NPROW = 2, NPCOL = 4, NB = 18
Решение системы (в файле OUTFILE)
X = (1.D0, 2.D0, 3.D0, . . . ,6999.D0, 7000.D0)

1. *Антонов А.С.* Параллельное программирование с использованием технологии MPI: Учебное пособие: М.: Изд-во МГУ. 2004. 71 с.
2. *Антонов А.С.* Практический курс MPI. MPI: A Message-Passing Interface Standard (Version 1.1).
3. *Б*иблиотека численного анализа НИВЦ МГУ. http://num-anal.srcc.msu.ru/
4. *Воеводин В.В.* Математические модели и методы в параллельных процессах. М.: Наука. 1986. 296 с.
5. *Воеводин В.В.* Математические основы параллельных вычислений. М.: Изд-во МГУ. 1991. 345 с.
6. *Воеводин В.В.* Параллельные структуры алгоритмов и программ. М.: ОВМ АН СССР. 1987. 148 с.
7. *Воеводин В.В., Воеводин Вл.В.* Параллельные вычисления. С.П. "БХВ-Петербург". 2002. 608 с.
8. *Воеводин В.В., Кузнецов Ю.А.* Матрицы и вычисления. М.: Наука. 1984. 320 с.
9. *Воеводин В.В.* Вычислительные основы линейной алгебры. М.: Наука. 1977. 304 с.
10. *Дацюк В.Н., Букатов А.А., Жегуло А.И.* Введение в организацию и методы программирования многопроцессорных вычислительных систем (методическое пособие, часть I). Ростов-на-Дону: Изд-во РГУ. 2000. 36 с.
11. *Дацюк В.Н., Букатов А.А., Жегуло А.И.* Среда параллельного программирования MPI (методическое пособие, часть II). Ростов-на-Дону: Изд-во РГУ. 2000. 65 с.
12. *Немнюгин С.А., Стесик О.Л.* "Параллельное программирование для многопроцессорных вычислительных систем". "БХВ-Петербург". 2002. 400 с.
13. *Корнеев В.Д.* "Параллельное программирование в MPI". Новосибирск: Изд-во СО РАН. 2000. 213 с.
14. *Арушанян О.Б., Волченскова Н.И.* Решение систем линейных алгебраических уравнений на распределенной памяти на основе пакета ScaLAPACK: Учебное пособие. (http://num-anal.srcc.msu.ru/prac_pos/page_6.htm)
15. *Арушанян О.Б., Волченскова Н.И.* Решение линейной алгебраической проблемы собственных значений для симметричных и эрмитовых матриц на распределенной памяти на основе пакета ScaLAPACK: Учебное пособие (готовится к публикации).
16. *Anderson E., Bai Z., Bischof C., Demmel J., Dongarra J., Du Croz J., Greenbaum A., Hammarling S., Mckenney A., Ostrouchov S. and Sorensen D.* LAPACK Users' Guide, Society for Industrial and Applied Mathematics. Philadelphia, PA, second

ed. 1995.
17. *Blackford L. S., Choi J., Cleary A., Demmej J., Dhillon I., Dongarra J.J., Hammarling S., Henry G., Petitet A., Stanley K., Walker D.W. and Whaley R. C.* ScaLAPACK: A portable linear algebra library for distributed memory computers — design issues and performance, in Proceedings of Supercomputing '96, Sponsored by ACM SIGARCH and IEEE Computer Society, 1996. (ACM Order Number: 415962, IEEE Computer Society Press Order Number: RS00126. http://www.supercomp.org/sc96/proceedings/).
18. http://www.netlib.org/scalapack/index.html/.
19. *Choi J., Dongarra J. and Walker D.* PB–BLAS: A Set of Parallel Block Basic Linear Algebra Subroutines, Practice and Experience. 8 (1996). pp. 517–535.
20. http://www.netlib.org/scalapack/html/pblas_qref.html/.
21. *Dongarra J., R. van de Geun and Whaley R.C.* Two dimensional basic linear algebra communication subprograms, in Environments and Tools for Parallel Scientific Computing, Advances in Parallel Computing, J. Dongarra and B. Tourancheau, eds., vol. 6, Elsevier Science Publishers B.V. 1993. pp. 31–40.
22. http://www.netlib.org/blacs/
23. *Golub G. and C.F. Van Loan.* Matrix Computations, Johns Hopkins University Press, Baltimore, MD, third ed. 1996.
24. *Lawson C.L., Hanson R.J., Kincaid D. and Krogh F.T.* Basic linear algebra subprograms for Fortran usage, ACM Trans. Math. Soft. 5 (1979). pp. 308–323.
25. http://www.netlib.org/blas/
26. *Lichtenstein W. and Johnsson S.L.* Block–cyclic dense linear algebra, SIAM J. Sci. Stat. Comput. 14 (1993). pp. 1259–1288.
27. *Prylli L. and Tourancheau B.* Efficient block cyclic data redistribution, in EUROPAR'96, vol. 1 of Lecture Notes in Computer Science, Springer-Verlag. 1996. pp. 155–165.
28. *Fernando K.V. and Parlett B.N.* Accurate singular values and differential qd algorithms// Numer.Math. 1994. Vol–67, № 2. pp. 191–230.
29. *Форсайт Дж., Малькольм М., Моулер К.* Машинные методы математических вычислений. М.: Изд-во МИР. 1980.