ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 108 за 2016 г.

ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

Суков С.А.

Библиотека препроцессорной обработки неструктурированных сеток hm4PreprocessorLib

Рекомендуемая форма библиографической ссылки: Суков С.А. Библиотека препроцессорной обработки неструктурированных сеток hm4PreprocessorLib // Препринты ИПМ им. М.В.Келдыша. 2016. № 108. 20 с. doi:10.20948/prepr-2016-108
URL: http://library.keldysh.ru/preprint.asp?id=2016-108

О р д е н а Л е н и н а
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В.Келдыша
Р о с с и й с к о й а к а д е м и и н а у к

С.А.Суков

Библиотека препроцессорной обработки неструктурированных сеток hm4PreprocessorLib

Москва — 2016

Суков С.А.

Библиотека препроцессорной обработки неструктурированных сеток hm4PreprocessorLib

В работе приводится описание библиотеки подпрограмм hm4PreprocessorLib. Библиотека включает в себя реализации стандартных процедур обработки топологии нерегулярных сеток и графов на этапе инициализации данных распределенных вычислительных экспериментов. Представлены результаты обработки сетки, содержащей $230 \cdot 10^6$ элементов типа тетраэдр, призма, пирамида и гексаэдр с запуском 336 параллельных процессов.

Ключевые слова: параллельные вычисления, численное моделирование, неструктурированные гибридные сетки

Sergey Alexandrovich Sukov

Unstructured mesh preprocessing library hm4PreprocessorLib

This paper contains the description of the parallel library hm4PreprocessorLib. The library includes routines for the unstructured meshes and graphs processing during initialization of parallel computing experiments. We present the results of the mesh processing, comprising $230 \cdot 10^6$ elements such as a tetrahedron, a prism, a pyramid and a hexahedron with the launch of 336 MPI-processes.

Key words: parallel computing, numerical modeling, unstructured grids

Работа выполнена при поддержке Российского фонда фундаментальных исследований, проекты №№ 15-07-04213-а, 14-07-00712-а.

Оглавление

1. Введение

Метод геометрического параллелизма — основной подход к разработке параллельных реализаций численных алгоритмов решения задач гидро- и газовой динамики, основанных на использовании метода конечного объема. Распределение вычислений внутри группы взаимодействующих в рамках модели передачи сообщений процессов происходит путем декомпозиции расчетной области на домены. Число доменов соответствует числу параллельных процессов, каждый из которых определяет актуальные значения искомых параметров в ячейках одного домена. Программная реализация чередует этапы вычислений и обменов данными. Синхронизация параллельных процессов обусловлена зависимостью между значениями сеточных функций в ячейках, относящихся к различным доменам. Число точек синхронизации и объемы передаваемых данных зависят от вычислительного шаблона, логической структуры алгоритма и стратегии минимизации накладных расходов.

Теоретический предел масштабируемости алгоритмов на основе метода геометрического параллелизма равен числу расчетных ячеек. Практическая параллельная эффективность и предел параллелизма программного обеспечения, а также возможность его использования для проведения широкомасштабных вычислительных экспериментов зависят от эффективности применяемых методов декомпозиции расчетной области и подходов к хранению и обработке больших объемов данных. В условиях доступа к ресурсам суперкомпьютеров через систему управления пуском задач практическое преимущество получают приложения с распределенным препроцессором исходных данных. Их плюс состоит, как минимум, в готовности к запуску на доступном в конкретный момент времени числе устройств. Подход с предварительной обработкой данных отдельной (возможно, последовательной) программой и записью конфигурационных файлов для каждого из предполагаемых вариантов запуска основного приложения, очевидно, менее эффективен.

Препроцессорный модуль включает в себя процедуры декомпозиции расчетной области, выделения геометрии расчетных подобластей процессов из топологического описания сетки в целом и инициализации схемы и структур приема/передачи данных. Проблема декомпозиции области обычно сводится к разбиению эквивалентного сетке контрольных объемов графа. Программные реализации соответствующих методов [1-4] сравнительно эффективно работают с графами, содержащими до 10^8 вершин. А следовательно, и остальные компоненты препроцессорного модуля должны поддерживать обработку сеток с сопоставимым числом элементов.

В данной работе приводится описание состава и функциональных возможностей библиотеки подпрограмм hm4PreprocessorLib. Библиотека hm4PreprocessorLib включает в себя программные реализации решений

типовых задач обработки неструктурированных сеток на этапе инициализации данных распределенных вычислительных экспериментов. В состав hm4PreprocessorLib не входят подпрограммы декомпозиции графов. Библиотечные процедуры строят дуальный граф сеток с элементами четырех типов и огрубленный граф декомпозиции дуального графа, выделяют и группируют геометрию элементов, принадлежащих расчетным областям процессов, инициализируют параметры схемы обменов данными.

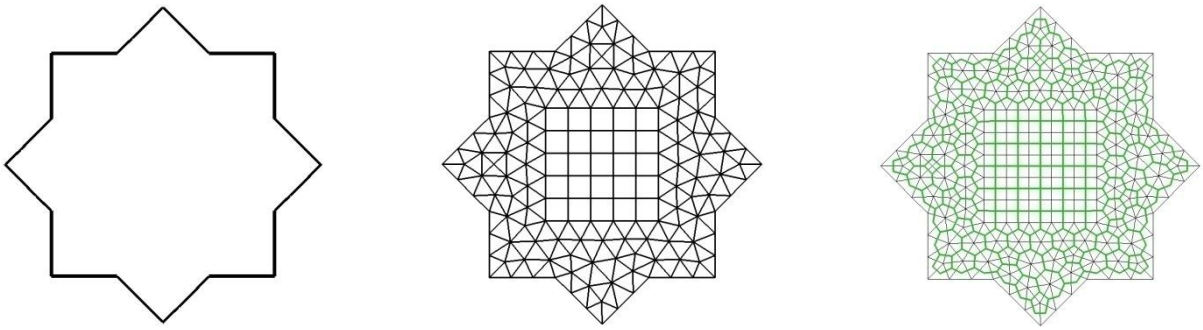
2. Базовый алгоритм организации распределенных вычислений

Функциональный состав библиотеки подпрограмм hm4PreprocessorLib сформирован исходя из описанного ниже варианта параллельной реализации явных численных методов расчета задач гидро- и газовой динамики.

Внутри модели расчетной области (рис. 1а) строится сетка контрольных объемов (рис. 1б). Сетка состоит из многогранных ячеек C_i, имеющих объем $|C_i|$ и поверхность ∂C_i. Поверхность ∂C_i внутренней ячейки представляет собой совокупность плоских граней ∂C_{ij} общих с ячейками множества I_i. Дискретные значения сеточных функций Q_i относятся к центрам масс контрольных объемов. Алгоритм вычисления значений сеточных функций на новом слое по времени $\bar{Q}_i$ во внутренней сеточной ячейке описывается формулой

$$\bar{Q}_i = Q_i - \frac{\Delta t}{|C_i|} \sum_{j \in I_i} H(Q_i, Q_j, \vec{n}_{ij})\, S_{ij}, \qquad (1)$$

где H — функция вычисления потока, $\vec{n}_{ij}$ — единичная нормаль к грани ∂C_{ij}, S_{ij} — площадь грани и Δt — шаг интегрирования по времени.



а) расчетная область

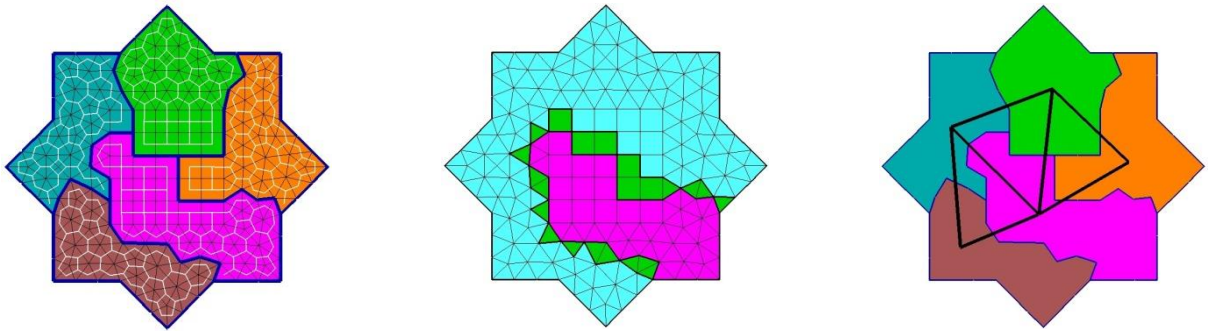
б) сетка контрольных объемов

в) сетка и ее дуальный граф

Рис. 1. Расчетная область, сетка и дуальный граф.

Параллельный расчет задачи выполняется группой из N_{proc} процессов. Синхронизация и взаимодействие процессов происходят в рамках модели передачи сообщений. Каждый процесс группы имеет целочисленный

идентификатор $P = 0,1,\dots,N_{proc} - 1$. Для декомпозиции сетки на N_{proc} доменов строится дуальный граф (рис. 1в). Его вершины соответствуют контрольным объемам, а ребра отражают соседство двух расчетных ячеек через общую грань. Дисбаланс вычислительной нагрузки учитывается в весовых коэффициентах вершин и ребер графа. Стандартная цель разбиения – получить сбалансированные по весу домены, одновременно минимизировав суммарный вес разрезанных ребер. Пример декомпозиции области для $N_{proc} = 5$ показан на рис. 2а.



а) разбиение на 5 доменов б) домен и буферная зона в) коммуникационный граф

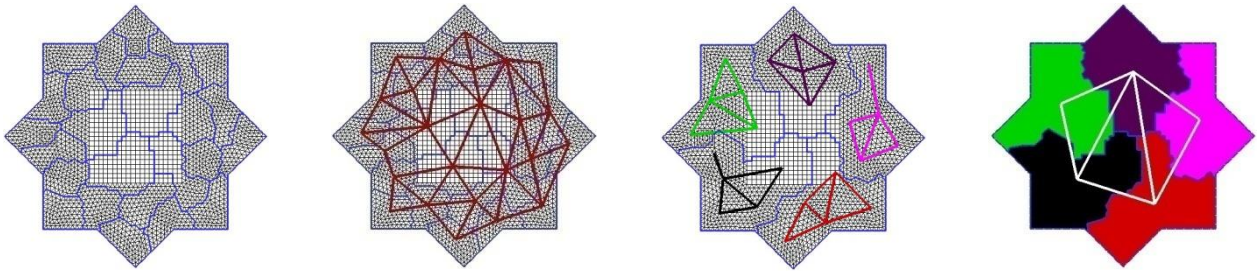
Рис. 2. Прямая декомпозиция расчетной области.

При любом разбиении для вычисления значений $\bar{Q}_i$ в приближенных к границе домена ячейках потребуются значения сеточных функций в контрольных объемах, относящихся к доменам других процессов. Множество таких ячеек формирует буферную зону домена. В соответствии с формулой (1) в нее попадают ячейки, у которых есть общая грань с принадлежащими домену ячейками. Расчетная область процесса представляет собой объединение домена и буферной зоны (рис. 2б). На каждом шаге интегрирования по времени процесс принимает значения сеточных функций в элементах буферной зоны и передает значения в ячейках своего домена, попадающих в буферные зоны других процессов. Описание схемы обмена данными включает в себя граф связей процессов (рис. 2в) и списки передаваемых значений.

Для декомпозиции сеток с большим числом элементов возможно применение алгоритма двухуровневого разбиения [5,6]. Первичная декомпозиция области состоит из $K_{deco}N_{proc}$ доменов, где $K_{deco} > 1$ — целочисленный коэффициент. На рис. 3а показан пример первичной декомпозиции области при $N_{proc} = 5$ и $K_{deco} = 5$.

На первичной декомпозиции строится огрубленный взвешенный граф (рис. 3б). Метод его построения аналогичен методу инициализации графа обменов данными для группы $K_{deco}N_{proc}$ процессов. Веса вершин огрубленного графа равны сумме весов принадлежащих соответствующему домену вершин дуального графа, веса ребер — сумме весов ребер дуального графа между принадлежащими двум доменам вершинами. Размер огрубленного графа в несколько раз меньше размера дуального графа. Поэтому повторная

декомпозиция огрубленного графа на N_{proc} частей выполняется в последовательном режиме. Результат декомпозиции огрубленного графа проецируется на дуальный граф (рис. 3в), после чего инициализируется схема обменов данными (рис. 3г). При обработке графов, содержащих порядка 10^8 вершин, двухуровневый алгоритм демонстрирует большую стабильность в смысле качества результата декомпозиции по сравнению с прямым разбиением.



а) первичная декомпозиция | б) огрубленный граф | в) декомпозиция области по огрубленному графу | г) коммуникационный граф

Рис. 3. Двухуровневая декомпозиция расчетной области.

Препроцессорная обработка данных завершается локализацией геометрии расчетных областей. На данном этапе в оперативной памяти процесса группируются данные, которых достаточно для инициализации геометрических параметров расчета. Из распределенного описания топологии сетки в целом выделяется описание топологии элементов, оказавшихся в расчетной области процесса, составляется список индексов принадлежащих элементам узлов, заполняется массив их координат.

Проанализировав представленный выше, по сути стандартный, подход к инициализации данных, можно выделить следующие процедуры, связанные с обработкой сеточной топологии и производных от нее графов:

- построение дуального графа;
- построение огрубленного графа на заданном разбиении;
- составление списков индексов, принадлежащих доменам сеточных элементов;
- инициализация буферных зон доменов;
- инициализация графа и структур приема/передачи данных;
- выделение топологического описания заданного множества сеточных элементов;
- инициализация массива координат заданного множества сеточных узлов.

Рассматриваемый подход к препроцессорной обработке данных формально соответствует реализациям численных методов первого порядка точности по пространству. Вычислительный шаблон элемента включает только соседей первого уровня связности по дуальному графу. Подготовка данных для расчета по схеме повышенного порядка точности отличается критерием формирования

буферных зон. В случае, когда шаблон состоит из элементов нескольких уровней связности по дуальному графу, буферная зона может быть определена путем многократного вызова процедуры поиска соседей первого уровня связности. Альтернативный вариант инициализации буферной зоны той же процедурой состоит в замене описания дуального графа сетки на списки индексов принадлежащих шаблонам элементов.

Переход к внутренней индексации сеточных элементов и узлов расчетных областей, выделение памяти и инициализация указателей на границы массивов приема/передачи сообщений выполняются уже на локализованном описании области в последовательном режиме. Оптимальный алгоритм перехода к внутренней индексации должен учитывать особенности реализации конкретного вычислительного ядра. Таким образом, включение в состав библиотеки реализации некоторого универсального алгоритма решения данной задачи не является целесообразным.

3. Форматы представления данных

Библиотечные функции работают с последовательным и распределенным форматами представления топологии нерегулярных графов и сеток, за основу которых взят формат сжатого хранения разреженных матриц (CSR – compressed sparse row). Распределенный формат описывает вариант декомпозиции последовательного представления на блоки для хранения данных в оперативной памяти группы процессов. Подаваемые на вход подпрограмм библиотеки hm4PreprocessorLib и генерируемые в результате их выполнения структуры данных совместимы с форматами представления графов библиотек распределенной декомпозиции METIS, ParMETIS и GridSpiderPar. Индексация вершин графов, сеточных примитивов и ячеек массивов начинается с 0.

3.1. Представление графов

Топология графа, содержащего N_{vg} вершин и N_{eg} связывающих вершины графа ребер, задается двумя целочисленными массивами $xG[N_{vg}+1]$ и $aG[xG[N_{vg}]]$. Внутри графа вершины идентифицируются целочисленными индексами $id_v = 0,1,\dots,N_{vg}-1$. Индексы соседей первого уровня связности вершины id_v или вершин графа, связанных ребрами с id_v, хранятся в массиве aG, начиная с элемента $aG[xG[id_v]]$ и заканчивая элементом $aG[xG[id_v+1]-1]$ включительно. То есть локальные списки соседей вершин графа объединяются в порядке возрастания их индексов в массив aG. Массив xG ($xG[0]=0$) используется для хранения границ списков. Число соседей вершины id_v равно $xG[id_v+1]-xG[id_v]$. В описании неориентированного графа ребра дублируются: (id_{v1}, id_{v2}) в списке соседей вершины id_{v1} и (id_{v2}, id_{v1}) в списке соседей вершины id_{v2}. Кроме того, для неориентированного графа выполняются равенства $xG[N_{vg}] = 2N_{eg}$ или $N_{eg} = xG[N_{vg}]/2$.

При необходимости учета весов вершин и/или ребер графа в его описание добавляются целочисленные массивы $wvG[N_{vg}]$ и $weG[xG[id_v]]$. Элемент массива $wvG[id_v]$ содержит вес вершины id_v. Вес ребра $(id_v, aG[j])$, где $j \in [xG[id_v], xG[id_v + 1] - 1]$, записывается в ячейку $weG[j]$. Как и в случае с топологией ребер, веса ребер неориентированного графа повторяются в массиве weG дважды.

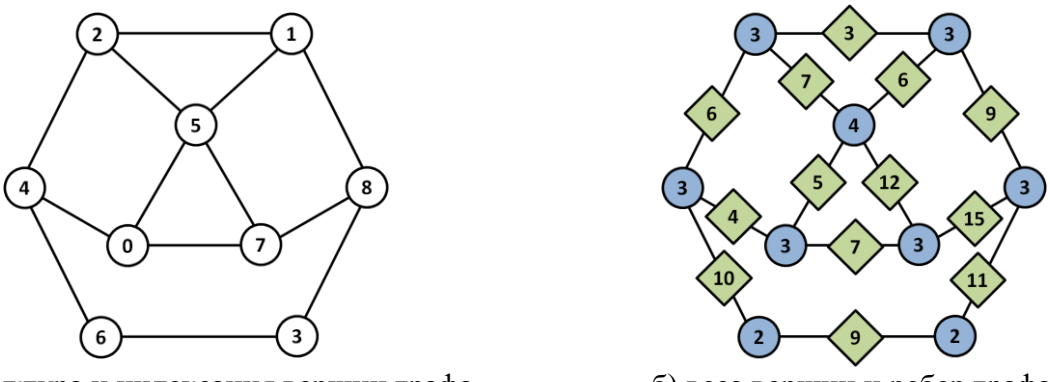
Декомпозиция данных распределенного представления соответствует линейному разбиению диапазона индексов вершин графа на N_{proc} интервалов. Распределение вершин по интервалам отображается в массиве $vgDist[N_{proc} + 1]$, где $vgDist[0] = 0$ и $vgDist[N_{proc}] = N_{vg}$. Параллельный процесс с идентификатором P хранит данные, которые связаны с $N_{vg}^P = vgDist[P + 1] - vgDist[P]$ вершинами P - го интервала с индексами $id_v \in [vgDist[P], vgDist[P + 1] - 1]$. Копии частей полного представления графа хранятся в массивах $wvG^P[N_{vg}^P]$, $aG^P[xG[vgDist[P + 1]] - xG[vgDist[P]]]$, и $weG^P[xG[vgDist[P + 1]] - xG[vgDist[P]]]$:

$$aG^P[\ \ ] = \{aG[xG[vgDist[P]]], aG[xG[vgDist[P]] + 1], \dots, aG[xG[vgDist[P + 1]] - 1]\},$$

$$weG^P[\ \ ] = \{weG[xG[vgDist[P]]], weG[xG[vgDist[P]] + 1], \dots, weG[xG[vgDist[P + 1]] - 1]\}.$$

Границы локальных списков соседей вершин задаются в массивах $xG^P[N_{vg}^P + 1]$: $xG^P[j] = xG[vgDist[P] + j] - xG[vgDist[P]]$, $j \in [0; N_{vg}^P - 1]$.

Таким образом, в рамках распределенного представления данных список соседей вершины графа $vgDist[P] \le id_v < vgDist[P + 1]$ хранится в памяти процесса P. Индексы соседних вершине элементов содержатся в массиве aG^P, начиная с элемента $aG^P[xG^P[id_v - vgDist[P]]]$ и заканчивая элементом $aG^P[xG^P[id_v + 1 - vgDist[P]] - 1]$ включительно. Аналогичным образом в массиве weG^P размещаются веса опирающихся на id_v ребер. Вес вершины записывается в ячейку массива $wvG^P[id_v - vgDist[P]]$.



а) структура и индексация вершин графа б) веса вершин и ребер графа

Рис. 4. Неориентированный взвешенный граф.

На рис. 4а показан пример неориентированного графа, содержащего 9 вершин и 13 ребер. Вес вершины графа равен числу опирающихся на эту вершину ребер. Вес ребра равен сумме индексов его вершин (рис. 4б). Ниже

дано схематичное описание топологии графа в последовательном и распределенном форматах.

xG	0 3 6 9 11 14 18 20 23 26
aG	4 5 7, 2 5 8, 1 4 5, 6 8, 0 2 6, 0 1 2 7, 3 4, 0 5 8, 1 3 7
wvG	3 3 3 2 3 4 2 3 3
weG	4 5 7, 3 6 9, 3 6 7, 9 11, 4 6 10, 5 6 7 12, 9 10, 7 12 15, 9 11 15

Описание графа в последовательном формате

$vgDist^0$	0 3 6 9
xG^0	0 3 6 9
aG^0	4 5 7, 2 5 8, 1 4 5
wvG^0	3 3 3
weG^0	4 5 7, 3 6 9, 3 6 7

$vgDist^1$	0 3 6 9
xG^1	0 2 5 9
aG^1	6 8, 0 2 6, 0 1 2 7
wvG^1	2 3 4
weG^1	9 11, 4 6 10, 5 6 7 12

$vgDist^2$	0 3 6 9
xG^2	0 2 5 8
aG^2	3 4, 0 5 8, 1 3 7
wvG^2	2 3 3
weG^2	9 10, 7 12 15, 9 11 15

Описание графа в распределенном формате ($N_{proc} = 3$)

3.2. Декомпозиция графов

Декомпозиция графа на N_{domen} частей (или "раскраска" вершин графа по доменам) описывается целочисленным массивом $part[N_{vg}]$. Ячейка массива $0 \le part[id_v] \le N_{domen}$ содержит индекс домена, которому принадлежит вершина графа id_v. Алгоритм распределенного хранения элементов массива part аналогичен алгоритму распределенной записи массива весов вершин wvG.

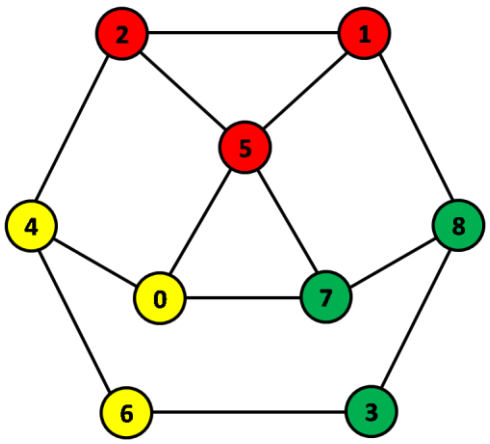


Рис. 5. Декомпозиция графа.

part	1 0 0 2 1 0 1 2 2

Последовательный формат

$vgDist^0$	0 3 6 9
$part^0$	1 0 0

$vgDist^1$	0 3 6 9
$part^1$	2 1 0

$vgDist^2$	0 3 6 9
$part^2$	1 2 2

Распределенный формат

На рис. 5 показан пример декомпозиции графа (рис. 4) на три части. Справа от рисунка приводится запись "раскраски" графа в последовательном и распределенном ($N_{proc} = 3$) форматах.

3.3. Схема обменов данными

Схема обмена данными P-го процесса группы описывается в четырех целочисленных массивах $xRecv[N_{proc} + 1]$, $xSend[N_{proc} + 1]$, $aRecv[xRecv[N_{proc}]]$ и $aSend[xSend[N_{proc}]]$. Число векторов значений сеточных функций, которые принимаются процессом P от процесса K, вычисляется как разность $N_{Recv}^{PK} = xRecv[K + 1] - xRecv[K]$. Глобальные индексы соответствующих элементов хранятся в ячейках массива aRecv, начиная с $aRecv[xRecv[K]]$ по $aRecv[xRecv[K + 1] - 1]$ включительно. Равенство $N_{Recv}^{PK} = 0$ означает, что процесс P не запрашивает данные от процесса K ($N_{Recv}^{PP} = 0$). Схема передачи сообщений отображается в массивах xSend и aSend, инициализация которых происходит по тому же принципу.

Схемы обменов данными различных процессов согласованны между собой. То есть в случае $N_{Recv}^{PK} > 0$ всегда выполняется равенство $N_{Recv}^{PK} = N_{Send}^{KP}$, где N_{Send}^{KP} — число векторов значений сеточных функций, которые передаются процессом K процессу P. И одновременно $aRecv^{P}[xRecv^{P}[K] + j] = aSend^{K}[xSend^{K}[P] + j]$, при $j < N_{Recv}^{PK}$.

Ниже приводится пример инициализации структур приема/передачи данных для рассматриваемого случая декомпозиции графа (рис. 5).

$xRecv^0$	0 0 2 4
$aRecv^0$	0 4, 7 8
$xSend^0$	0 0 2 4
$aSend^0$	2 5, 1 5

$xRecv^1$	0 2 2 4
$aRecv^1$	2 5, 3 7
$xSend^1$	0 2 2 4
$aSend^1$	0 4, 0 6

$xRecv^2$	0 2 4 4
$aRecv^2$	1 5, 0 6
$xSend^2$	0 2 4 4
$aSend^2$	7 8, 3 7

Схема приема/передачи данных

3.4. Топология сетки

Описание сетки, содержащей N_{vm} узлов и N_{em} элементов, состоит из списков индексов образующих элементы узлов и массива с парами (сетка на плоскости) или тройками (пространственная сетка) их координат. Узлы и элементы сетки идентифицируются целочисленными индексами $id_{vm} = 0,1, \dots, N_{vm} - 1$ и $id_{em} = 0,1, \dots, N_{em} - 1$ соответственно. Метод записи топологии сеточных элементов аналогичен методу записи топологии нерегулярных графов. Типы элементов и списки индексов их узлов хранятся в массивах $xM[N_{em} + 1]$ и $aM[xM[N_{em}]]$. Здесь предполагается, что тип элемента id_{em} можно определить по числу его узлов, которое равно $xM[id_{em} + 1] - xM[id_{em}]$. Индексы принадлежащих элементу узлов хранятся в массиве aM, начиная с

элемента $aM[xM[id_{em}]]$ и заканчивая элементом $aM[xM[id_{em}+1]-1]$ включительно.

Плоские или пространственные координаты группируются по узлам и хранятся в массиве $cM[N_{vm} \cdot N_{dim}]$, где $N_{dim} = 2,3$ — размерность пространства. То есть, например, при $N_{dim} = 3$ координаты узла определяются как $(x_{id_{vm}}, y_{id_{vm}}, z_{id_{vm}}) = (cM[id_{vm} \cdot N_{dim}], cM[id_{vm} \cdot N_{dim}+1], cM[id_{vm} \cdot N_{dim}+2])$.

В распределенном виде массивы aM и cM делятся на блоки в соответствии с линейным разбиением на N_{proc} интервалов диапазонов индексов элементов и узлов сетки. Границы интервалов фиксируются в массивах $emDist[N_{proc}+1]$ и $vmDist[N_{proc}+1]$. Список индексов принадлежащих элементу $emDist[P] \le id_{em} < emDist[P+1]$ узлов располагается в памяти P-го процесса в ячейках массива aM^P с $aM^P[xM^P[id_{em} - emDist[P]]]$ по $aM^P[xM^P[id_{em}+1-emDist[P]]-1]$ включительно. Координаты узла $vmDist[K] \le id_{vm} < vmDist[K+1]$ хранятся K-м процессом в ячейках массива cM^K с $cM^K[(id_{vm} - vmDist[K]) \cdot N_{dim}]$ по $cM^K[(id_{vm} - vmDist[K]) \cdot N_{dim} + N_{dim} - 1]$.

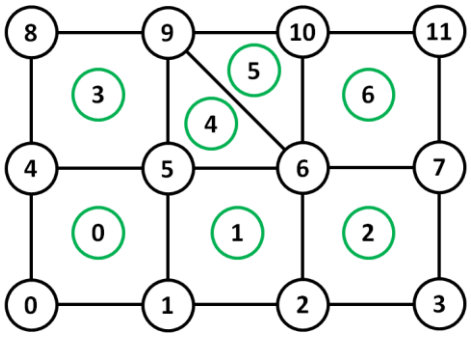


Рис. 6. Сетка на плоскости с индексацией узлов и элементов.

На рис. 6 показана сетка, состоящая из треугольных и четырехугольных элементов. Ниже приводится пример записи топологии этой сетки в последовательном и распределенном ($N_{proc} = 3$) форматах.

xM	0 4 8 12 16 19 22 26
aM	0 1 4 5, 1 2 5 6, 2 3 6 7, 4 5 8 9, 5 6 9, 9 6 10, 10 6 11 7
cM	0 0, 1 0, 2 0, 3 0, 0 1, 1 1, 2 1, 3 1, 0 2, 1 2, 2 2, 3 2

Запись сеточной топологии в последовательном формате

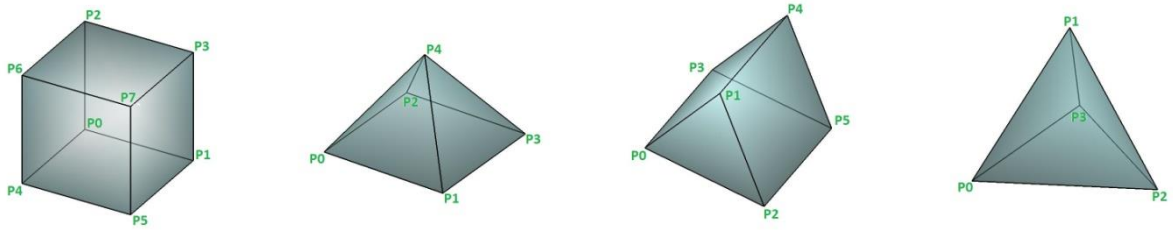
$emDist^0$	0 3 5 7
xM^0	0 4 8 12
aM^0	0 1 4 5, 1 2 5 6, 2 3 6 7
$vmDist^0$	0 4 8 12
cM^0	0 0, 1 0, 2 0, 3 0

$emDist^1$	0 3 5 7
xM^1	0 4 7
aM^1	4 5 8 9, 5 6 9
$vmDist^1$	0 4 8 12
cM^1	0 1, 1 1, 2 1, 3 1

$emDist^2$	0 3 5 7
xM^2	0 3 7
aM^2	9 6 10, 10 6 11 7
$vmDist^2$	0 4 8 12
cM^2	0 2, 1 2, 2 2, 3 2

Запись сеточной топологии в распределенном формате

Для корректной инициализации библиотечной функцией дуального графа сетки следует придерживаться показанного на рис. 7 алгоритма перечисления индексов узлов гексаэдров, четырехугольных пирамид, треугольных призм и тетраэдров. Стандартизация алгоритма перечисления узлов в описании элемента необходима для правильного определения топологии его граней.



а) гексаэдр
P0 P1 P2 P3 P4 P5 P6 P7

б) пирамида
P0 P1 P2 P3 P4

в) призма
P0 P1 P2 P3 P4 P5

г) тетраэдр
P0 P1 P2 P3

Рисунок 7. Алгоритм перечисления индексов узлов.

4. Библиотека hm4PreprocessorLib

Полный перечень функций, входящих в состав библиотеки hm4PreprocessorLib, с кратким описанием их назначения приводится в табл. 1.

Таблица 1

Подпрограмма	Назначение
hm4MeshDualGraphMPI	Инициализация дуального графа
hm4CoarseGraphMPI	Инициализация огрубленного графа
hm4DomenElementsListMPI	Составление списка индексов элементов обрабатываемого домена
hm4DualLinksL1MPI	Инициализация списка индексов элементов первого уровня связности
hm4RecvMessageStructuresMPI	Инициализация схемы приема данных
hm4SendMessageStructuresMPI	Инициализация схемы передачи данных
hm4DomenElementsTopologyMPI	Инициализация топологии элементов расчетной области
hm4DomenNodesMPI	Инициализация списка индексов и координат узлов расчетной области

Препроцессорная обработка данных начинается с вызова подпрограммы **hm4MeshDualGraphMPI**, которая инициализирует распределенный дуальный граф гибридных сеток с элементами типа тетраэдр, треугольная призма, четырехугольная пирамида и гексаэдр. Дуальный граф подается на вход сторонней программе декомпозиции, генерирующей массив с "раскраской"

вершин графа по заданному числу доменов. При использовании двухуровневой схемы разбиения генерация огрубленного взвешенного графа по промежуточному разбиению осуществляется библиотечной функцией **hm4CoarseGraphMPI.** Огрубленный граф снова подается на вход сторонней программе. Окончательное разбиение огрубленного графа проецируется на дуальный граф.

Списки индексов элементов, вошедших в обрабатываемый процессом домен, составляются подпрограммой **hm4DomenElementsListMPI.** Подпрограмма **hm4DualLinksL1MPI** вызывается для определения списков индексов элементов первого уровня связности, попадающих в буферные зоны доменов. Далее списки передаются в качестве входных параметров подпрограмме **hm4SendMessageStructuresMPI**, которая отвечает за генерацию схемы приема данных. Согласованная с ней схема передачи данных генерируется подпрограммой **hm4SendMessageStructuresMPI.**

Подпрограмма **hm4DualLinksL1MPI** поддерживает возможность определения буферной зоны на ориентированном графе, где существование ребра $(\mathrm{id}_{v1}, \mathrm{id}_{v2})$ не равносильно существованию ребра $(\mathrm{id}_{v2}, \mathrm{id}_{v1})$. Таким образом, подпрограмма может использоваться для построения буферных зон на приведенном к CSR-формату описании шаблонов сеточных элементов. То есть для инициализации буферных зон в реализациях методов повышенного порядка точности по пространству.

Расчетная область процесса включает в себя элементы домена и его буферной зоны. Указатель на объединенный список их индексов передается в качестве входного параметра подпрограммы **hm4DomenElementsTopologyMPI,** которая копирует в оперативную память процесса описание топологии запрашиваемых элементов. Список индексов попавших в расчетную область процесса узлов и массив их координат генерируются подпрограммой **hm4DomenNodesMPI.**

Процедура **hm4DomenElementsTopologyMPI** обрабатывает списки узлов произвольной длины. Однако принятый формат описания топологии элементов в явном виде не содержит идентификаторов их типов. Для гибридных сеток, состоящих из тетраэдров, гексаэдров, призм и пирамид, тип элемента однозначно определяется числом его вершин. Но для сеток с совпадающими по числу узлов элементами разных типов (например, призмами и октаэдрами) такой подход уже не применим. Для обработки библиотечными функциями топологии произвольных сеток целочисленный идентификатор типа элемента должен быть скрыт в списке индексов его узлов. После вызова функции **hm4DomenElementsTopologyMPI** идентификатор следует удалить из списка, а затем выполнить процедуру инициализации координат узлов **hm4DomenNodesMPI.**

4.1. Прототипы и вызов библиотечных функций

Исходный текст программных модулей написан на языке C. Взаимодействие параллельных процессов реализовано с использованием интерфейса передачи сообщений MPI. Вызов библиотечных функций выполняется одновременно всеми процессами группы (MPI_Comm comm). При этом подпрограммы определения геометрии расчетных областей инициализируют массивы, соответствующие только домену, идентификатор которого совпадает с идентификатором процесса (MPI_Comm_rank). Внутреннее распределение памяти осуществляется функцией malloc.

В случае успешного завершения все библиотечные функции возвращают значение 0. При возникновении ошибок в ходе работы подпрограмм происходит аварийный выход из приложения (MPI_Abort(comm, 0)) с выводом в поток stderr информационного сообщения с описанием ошибки.

Параметр outSTREAM задает поток вывода информационных сообщений библиотечными функциями. В случае outSTREAM = NULL информационные сообщения на консоль не выводятся.

Далее приводятся прототипы библиотечных функций.

int **hm4MeshDualGraphMPI**(int *emDist, int *xM, int *aM, int **xG, int **aG,
FILE *outSTREAM, MPI_Comm comm)

Описание Подпрограмма инициализации распределенного дуального графа сетки, состоящей из тетраэдров, треугольных призм, четырехугольных пирамид и гексаэдров.

Параметры

emDist — Массив с описанием линейного распределения сеточных элементов по процессам группы comm.

xM, aM — Указатели на массивы распределенного описания топологии сеточных элементов.

xG, aG — Адреса переменных, в которые записываются указатели на массивы с распределенным описанием дуального графа. Линейное распределение вершин графа по процессам соответствует emDist.

int **hm4CoarseGraphMPI** (int *vgDist, int *xG, int *aG, int nPart, int *part, int **cxG, int **caG,
int **wghtVCG, int **wghtECG, FILE *outSTREAM, MPI_Comm comm)

Описание Инициализация огрубленного графа с целочисленными весами вершин и ребер на заданном разбиении дуального графа. Структуры данных с последовательным описанием топологии огрубленного графа инициализируются только на нулевом процессе группы comm.

Параметры

vgDist — Массив с описанием линейного распределения вершин

дуального графа по процессам группы comm.

xG, aG — Массивы с распределенным описанием дуального графа. Линейное распределение вершин графа по процессам соответствует vgDist.

nPart — Число доменов в первичной декомпозиции дуального графа.

part — Распределенный в соответствии с vgDist массив с описанием первичного разбиения дуального графа на nPart доменов.

cxG, caG — Адреса переменных, в которые записываются указатели на массивы с последовательным описанием топологии огрубленного графа. Число вершин графа равно nPart.

wghtVCG — Адрес переменной, в которую записывается указатель на массив с целочисленными весами вершин огрубленного графа. Вес вершины огрубленного графа равен сумме весов вершин дуального графа, вошедших в соответствующий домен первичного разбиения. Вес вершины дуального графа равен числу исходящих из нее ребер. В случае wghtVCG = NULL веса не инициализируются.

wghtECG — Адрес переменной, в которую записывается указатель на массив с целочисленными весами ребер огрубленного графа. Вес ребра огрубленного графа равен числу ребер дуального графа, соединяющих принадлежащие соответствующим доменам первичного разбиения вершины. В случае wghtECG = NULL веса не инициализируются.

int **hm4DomenElementsListMPI** (int *vgDist, int *part, int *neProc, int **eProc, FILE *outSTREAM, MPI_Comm comm)

Описание — Инициализация для каждого из процессов группы comm списка индексов элементов, принадлежащих обрабатываемому процессом домену.

Параметры

vgDist — Массив с описанием линейного распределения вершин дуального графа по процессам группы comm.

part — Распределенный в соответствии с vgDist массив с описанием декомпозиции дуального графа на домены, число которых равно размеру группы comm.

neProc — Указатель на переменную, в которую записывается общее число принадлежащих домену элементов.

eProc — Адрес переменной, в которую записывается указатель на упорядоченный по возрастанию массив с глобальными индексами принадлежащих домену элементов.

int **hm4DualLinksL1MPI** (int *vgDist, int *xG, int *aG, int neProc, int *eProc, int *neL1, int **eL1, FILE *outSTREAM, MPI_Comm comm)

Описание Инициализация списков индексов элементов первого уровня связности.

Параметры

vgDist — Массив с описанием линейного распределения вершин ориентированного графа по процессам группы comm.

xG, aG — Массивы с распределенным описанием ориентированного графа. Линейное распределение вершин графа по процессам соответствует vgDist.

neProc — Число элементов домена.

eProc — Указатель на массив со списком индексов принадлежащих домену элементов.

neL1 — Указатель на переменную для записи общего числа элементов, находящихся в зоне первого уровня связности.

eL1 — Адрес переменной, в которую записывается указатель на массив с упорядоченными по возрастанию индексами элементов первого уровня связности.

int **hm4RecvMessageStructuresMPI** (int *vgDist, int *part, int neL1, int *eL1,
int *xRecv, int *aRecv, FILE *outSTREAM, MPI_Comm comm)

Описание Инициализация структур с описанием схемы приема данных процессами группы comm.

Параметры

vgDist — Массив с описанием линейного распределения вершин дуального графа по процессам группы comm.

part — Распределенный в соответствии с vgDist массив с описанием декомпозиции дуального графа.

neL1, eL1 — Число и указатель на массив со списком индексов элементов буферной зоны домена.

xRecv, aRecv — Указатели на предварительно распределенные пользователем целочисленные массивы для записи схемы приема данных параллельным процессом.

int **hm4SendMessageStructuresMPI** (int *xRecv, int *aRecv, int **xSend, int **aSend,
FILE *outSTREAM, MPI_Comm comm)

Описание Инициализация структур с описанием схемы передачи данных процессами группы comm.

Параметры

xRecv, aRecv — Схема приема данных.

xSend, aSend — Адреса переменных, в которые записываются указатели на массивы с описанием схемы передачи данных параллельным процессом.

int **hm4DomenElementsTopologyMPI** (int *emDist, int *xM, int *aM, int neAProc, int *eAProc,
int **xMP, int **aMP, FILE *outSTREAM, MPI_Comm comm)

Описание Выделение описания топологии элементов, принадлежащих расчетным областям процессов, из распределенного описания топологии сетки в целом.

Параметры

emDist — Массив с описанием линейного распределения сеточных элементов по процессам группы comm.

xM, aM — Распределенное в соответствии с emDist описание топологии сеточных элементов.

neAProc, eAProc — Число и указатель на массив со списком индексов элементов, принадлежащих расчетной области процесса.

xMP, aMP — Адреса переменных, в которые записываются указатели на массивы с описанием топологии элементов расчетной области. Списки узлов элементов отсортированы в соответствии с массивом eAProc.

int **hm4DomenNodesMPI** (int neAProc, int *xMP, int *aMP, int *vmDist, double *cM, int *nvAProc, int **vAProc, double **cMP, int nDim, FILE *outSTREAM, MPI_Comm comm)

Описание Инициализация списка индексов и массива координат сеточных узлов, находящихся в расчетной области процесса.

Параметры

neAProc — Число элементов в расчетной области процесса.

xMP, aMP — Массивы с описанием топологии элементов расчетной области.

vmDist — Массив с описанием линейного распределения координат сеточных узлов по процессам группы comm.

cM — Распределенный в соответствии с vmDist массив координат сеточных узлов.

nvAProc — Указатель на переменную для записи числа сеточных узлов, принадлежащих расчетной области процесса.

vAProc — Адрес переменной, в которую записывается указатель на массив с упорядоченными по возрастанию глобальными индексами сеточных узлов, принадлежащих расчетной области.

cMP — Адрес переменной, в которую записывается указатель на массив с координатами узлов расчетной области.

nDim — Размерность пространства.

4.2. Пример вызова библиотечных функций

Данный раздел содержит пример фрагмента исходного кода программы на языке C с вызовом библиотечных функций. Код соответствует реализации препроцессорной обработки данных для случая прямой декомпозиции сетки контрольных объемов.

```
////////////////////////////////////////////////////////////////////////////

// ПОСТРОЕНИЕ ДУАЛЬНОГО ГРАФА
```

```
hm4MeshDualGraphMPI(emDist, xM, aM, &xG, &aG, NULL, MPI_COMM_WORLD);

////////////////////////////
//  ДЕКОМПОЗИЦИЯ ГРАФА    //
////////////////////////////

// СОСТАВЛЕНИЕ СПИСКА ИНДЕКСОВ ОБРАБАТЫВАЕМЫХ ПРОЦЕССОМ ЭЛЕМЕНТОВ
hm4CreateProcessElementsListMPI(emDist, part, &neProc, &eProc, NULL, MPI_COMM_WORLD);

// ИНИЦИАЛИЗАЦИЯ СПИСКА ИНДЕКСОВ ЭЛЕМЕНТОВ БУФЕРНОЙ ЗОНЫ
hm4DualLinksL1MPI(emDist, xG, aG, neProc, eProc, &neL1, &eL1, NULL, MPI_COMM_WORLD);

// ОБЪЕДИНЕНИЕ СПИСКА ИНДЕКСОВ ЭЛЕМЕНТОВ ДОМЕНА СО СПИСКОМ ИНДЕКСОВ ЭЛЕМЕНТОВ
// БУФЕРНОЙ ЗОНЫ
eProc = (int *)realloc(eProc, (neL1 + neProc) * sizeof(int));
for(i=0; i<neL1; i++) eProc[neProc + i] = eLink[i];

// ИНИЦИАЛИЗАЦИЯ СХЕМЫ ПРИЕМА ДАННЫХ
xRecv = (int *)malloc( (Nproc + 1) * sizeof(int));
aRecv = (int *)malloc( neL1 * sizeof(int));
hm4RecvMessagesStructuresMPI(emDist, part, neL1, eProc + neProc, xRecv, aRecv,
                             NULL, MPI_COMM_WORLD);

// ИНИЦИАЛИЗАЦИЯ СХЕМЫ ПЕРЕДАЧИ ДАННЫХ
hm4SendMessageStructuresMPI(xRecv, aRecv, &xSend, &aSend, NULL, MPI_COMM_WORLD);

// ВЫДЕЛЕНИЕ ТОПОЛОГИИ ЭЛЕМЕНТОВ РАСЧЕТНОЙ ОБЛАСТИ
hm4DomenElementsTopologyMPI(emDist, xM, aM, neProc + neL1, eProc, &xMP, &aMP,
                            NULL, MPI_COMM_WORLD);

// ИНИЦИАЛИЗАЦИЯ СПИСКА ИНДЕКСОВ И КООРДИНАТ УЗЛОВ РАСЧЕТНОЙ ОБЛАСТИ
hm4DomenNodesMPI(neProc + neL1, xMP, aMP, vmDist, cM, &nvAProc, &vAProc, &cMP, 3,
                 NULL, MPI_COMM_WORLD);

////////////////////////////////////////////////////////////////////////////////
```

Фрагмент кода с примерами вызовов библиотечных функций.

5. Вычислительный эксперимент

Эффективность препроцессорного модуля в целом оценивается по двум основным критериям: сбалансированность распределения вычислительной нагрузки и ограничение на максимальный размер обрабатываемых сеток. Время инициализации данных вычислительного эксперимента не является основным показателем эффективности. Тем не менее, высокая скорость распределенной обработки топологии сетки и дуального графа имеет существенное значение при проведении широкомасштабных вычислительных экспериментов.

Для оценки корректности работы и быстродействия процедур библиотеки hm4PreprocessorLib разработанное программное обеспечение использовалось в отладочных версиях комплексов программ моделирования задач газовой динамики на неструктурированных сетках. В процессе тестирования, в частности, проводились расчеты течения в цилиндрической трубе на сетке, содержащей 135 457 921 узел и 230 243 456 элементов. Распределение сеточных элементов по типам дано в табл. 2. Объем занимаемого топологическим описанием дискового пространства составляет 9.3 ГБ. На рис. 8 показан вид сетки в срединном сечении расчетной области.

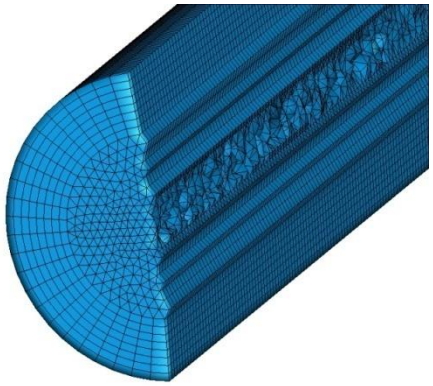


Рис. 8. Сетка в срединном сечении.

Таблица 2

Тип элемента	Число элементов
Тетраэдр	47 437 056
Призма	109 312 000
Пирамида	1 814 400
Гексаэдр	71 680 000
Всего	230 243 456

Времена выполнения библиотечных функций для случаев запуска 168 и 336 MPI-процессов приводятся в табл. 3. Запуск программы выполнялся на 14 и 28 модулях вычислительного комплекса К-100 (ИПМ им. М.В. Келдыша).

Таблица 3

Процедура	Время выполнения (секунд)	
	168 процессов	336 процессов
hm4MeshDualGraphMPI	5.69	4.33
hm4DomenElementsListMPI	7.52	5.12
hm4DualLinksL1MPI	23.88	20.69
hm4RecvMessageStructuresMPI	4.69	6.98
hm4SendMessageStructuresMPI	0.01	0.01
hm4DomenElementsTopologyMPI	33.77	41.60
hm4DomenNodesMPI	18.55	20.39
Общее время	94.11	99.12

Представленные в табл. 3 результаты можно трактовать как демонстрацию приемлемой скорости обработки существенных объемов сеточных данных при запуске большого числа параллельных процессов. Суммарное время выполнения подпрограмм находится в пределах 100 секунд. Очевидно, что существует зависимость между временем работы библиотечных функций и начальным распределением данных. Но даже в случае трехкратного снижения производительности инициализация данных, включая декомпозицию графа, займет не более 6 минут на фоне измеряемого уже часами времени выполнения одного сеанса вычислительного эксперимента. Таким образом, представленные результаты демонстрируют возможность эффективного использования разработанного алгоритмического и программного обеспечения при проведении широкомасштабных вычислительных экспериментов.

6. Список литературы

1. Головченко Е. Н., Якобовский М. В. Пакет параллельной декомпозиции больших сеток GridSpiderPar // Вычислительные методы и программирование, М.: МГУ. — 2015. — Т. 16, № 4. — С. 507–517.
2. Boman E., Devine K., Catalyurek U., Bozdag D., Hendrickson B., Mitchell W.F., Teresco J. Zoltan: Parallel Partitioning, Load Balancing and Data-Management Services. Developer's Guide, Version 3.3 // Sandia National Laboratories, Copyright © 2000-2010, URL: http://www.cs.sandia.gov/Zoltan/dev_html/dev.html.
3. Karypis George, Kumar Vipin. A Parallel Algorithm for Multilevel Graph Partitioning and Sparse Matrix Ordering. Journal of Parallel and Distributed Computing, 48, 1998, 71-95.
4. Pellegrini Francois. PT-Scotch and libScotch 5.1 User's Guide. Bacchus team, INRIA Bordeaux Sud-Ouest, ENSEIRB & LaBRI, UMR CNRS 5800, Universite Bordeaux I, Talence, France, 2010, 1- 78.
5. Суков С.А., Якобовский М.В. Обработка трехмерных неструктурированных сеток на многопроцессорных системах с распределенной памятью. / В сб. "Фундаментальные физико-математические проблемы и моделирование технико-технологических систем", вып. 6, под ред. Л.А. Уваровой. М., Изд-во "Janus-K", 2003, С. 233-239.
6. Суков С.А. Методы повышения эффективности широкомасштабных распределенных вычислительных экспериментов на неструктурированных сетках // Суперкомпьютерные дни в России: Труды международной конференции (28-29 сентября 2015 г., г. Москва). – М.: Изд-во МГУ. 2015. С. 269-272.