УДК 004.272

ЭВРИСТИЧЕСКИЕ АЛГОРИТМЫ ОТОБРАЖЕНИЯ ПАРАЛЛЕЛЬНЫХ MPI-ПРОГРАММ НА МУЛЬТИКЛАСТЕРНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ И GRID-СИСТЕМЫ

М. Г. Курносов[1], **А. А. Пазников**[2]

На основе методов разбиения графов предложен метод и эвристические алгоритмы отображения параллельных программ на распределенные вычислительные системы с иерархической структурой. Оптимизация достигается за счет распределения интенсивно обменивающихся ветвей параллельной программы по процессорным ядрам, связанным быстрыми каналами связи. В методе учитываются все иерархические уровни коммуникационной сети вычислительной системы. Приводятся результаты экспериментов по отображению MPI-программ из пакетов SPEC MPI и NAS Parallel Benchmarks на пространственно-распределенную мультикластерную вычислительную систему. Работа выполнена при поддержке РФФИ (гранты 11–07–00105, 12–07–31016, 10–07–00157, 12–07–31016), Минобрнауки РФ в рамках реализации Федеральной целевой программы “Научные и научно-педагогические кадры инновационной России” на 2009–2013 годы (грант 2012–1.1–12–000–1005–018) и Совета по грантам Президента РФ для поддержки ведущих научных школ (грант НШ–2175.2012.9).

Ключевые слова: отображение параллельных программ, пространственно-распределенные вычислительные системы, GRID-системы, параллельные вычисления, MPI.

Введение. Пространственно-распределенная вычислительная система (ВС) — это совокупность территориально рассредоточенных вычислительных подсистем (вычислительных кластеров и/или проприетарных систем с массовым параллелизмом), логически объединенных в единую систему для решения на ней параллельных задач [1]. К классу пространственно-распределенных ВС относятся мультикластерные и GRID-системы. Параллельные программы для таких ВС преимущественно разрабатываются в модели передачи сообщений, как правило, с использованием библиотек стандарта MPI (PACX [2], MPICH-G2 [3], GridMPI [4], MCMPI [5], Stampi [6], NumGRID [7]) или специализированных пакетов распределенных вычислений (X-Com [8], BOINC [9]).

Одной из важных проблем организации функционирования пространственно-распределенных ВС является оптимизация отображения на них параллельных программ (task mapping, task allocation, task assignment). Под оптимальным отображением понимается такое распределение ветвей параллельной программы по процессорным ядрам ВС, при котором достигается минимум накладных расходов на межмашинные обмены информацией.

Коммуникационные среды пространственно-распределенных ВС имеют иерархическую организацию. Как правило, в них можно выделить минимум три уровня:

первый уровень — сеть связи между подсистемами (кластерами);

второй уровень — сеть связи между вычислительными узлами отдельной подсистемы (кластера);

третий уровень — общая память вычислительных узлов.

Время передачи информации между процессорными ядрами существенно зависит от их размещения в системе [10].

Существующие библиотеки MPI (MPICH2, Open MPI, Intel MPI и др.) реализуют алгоритмы отображения параллельных программ с учетом только двух уровней коммуникационной сети — сети межузловых связей и общей памяти вычислительных узлов. Например, пакет hwloc [11, 12], который входит в состав библиотеки MPICH2 и Open MPI, использует информацию об иерархической структуре распределенных

[1] Институт физики полупроводников им. А. В. Ржанова Сибирского отделения РАН, просп. акад. Лаврентьева, 13, 630090, г. Новосибирск; Сибирский государственный университет телекоммуникаций и информатики, ул. Кирова, 86, 630102, г. Новосибирск; науч. сотр., доцент, e-mail: mkurnosov@gmail.com

[2] Институт физики полупроводников им. А. В. Ржанова Сибирского отделения РАН, просп. акад. Лаврентьева, 13, 630090, г. Новосибирск; Сибирский государственный университет телекоммуникаций и информатики, ул. Кирова, 86, 630102, г. Новосибирск; инженер-программист, аспирант, e-mail: apaznikov@gmail.com

ВС и реализует отображение при помощи библиотеки Scotch [13] многоуровневого разбиения графов. В работах [14–16] предложены алгоритмы отображения программ в системы, имеющие тороидальную структуру коммуникационной среды. Некоторые из предложенных алгоритмов [17] ориентированы на оптимизацию отображения MPI-программ определенного класса. В работе [18] исследуются универсальные эвристические алгоритмы отображения в гетерогенные системы, проводится сравнение производительности алгоритмов и их масштабируемости на большемасштабных системах. В качестве показателей эффективности используется средняя длина пути (в смысле теории графов), пройденная сообщением, и минимальное время передачи сообщения. В статье [19] рассматривается задача отображения виртуальных MPI-топологий на SMP-кластеры. При этом минимизируются накладные расходы на межмашинные обмены информацией.

Немаловажным является то, как в целевой функции (критерий оптимизации) учитывается интенсивность взаимодействия параллельных ветвей. Время выполнения программ с большим числом информационных обменов небольшого размера (это характерно для программ на языках семейства PGAS — Partition Global Address Space) существенно зависит от накладных расходов, связанных с латентностью при передаче информационных сообщений. В алгоритмах отображения таких программ целесообразно учитывать количество обменов, а не суммарный размер сообщений. Данное обстоятельство в недостаточной степени учитывается в существующих работах.

Применение известных методов отображения MPI-программ возможно и в пространственно-распределенных ВС, однако они не обеспечивают предельной эффективности использования ресурсов ВС, так как не учитывают все иерархические уровни коммуникационной среды. В пространственно-распределенных ВС важно учитывать наличие медленных каналов связи между подсистемами. Производительность этого уровня, как правило, значительно уступает остальным и существенным образом влияет на время выполнения параллельной программы.

В настоящей статье предлагаются эвристические алгоритмы отображения MPI-программ на пространственно-распределенные ВС. В предложенных алгоритмах учитываются все иерархические уровни коммуникационной сети ВС.

1. Отображение на пространственно-распределенные ВС. Пусть имеется пространственно-распределенная ВС, состоящая из H подсистем. Коммуникационная среда системы имеет иерархическую организацию и может быть представлена в виде дерева, содержащего L уровней.

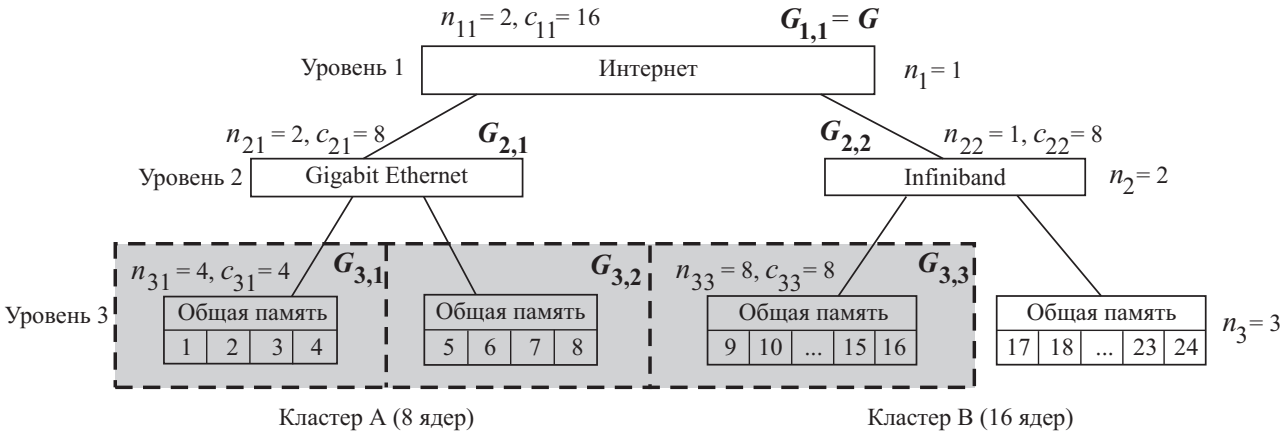

Рис. 1. Пример подсистемы ЭМ для решения параллельной задачи ранга $N = 16$

Рассмотрим (рис. 1) отображение параллельной программы на подсистему из N элементарных машин (ЭМ). Для параллельной программы системой управления ресурсами выделяется подсистема (на рис. 1 обозначена серым цветом), которая так же имеет иерархическую структуру. Введем обозначения: n_l — количество элементов на уровне $l \in \{1, 2, \ldots, L\}$; n_{lk} — количество прямых дочерних узлов элемента $k \in \{1, 2, \ldots, n_l\}$, находящегося на уровне l; c_{lk} — количество ЭМ, принадлежащих потомкам данного элемента.

Параллельная программа, созданная в модели передачи сообщений, может быть представлена информационным графом $G = (V, E)$, где $V = \{1, 2, \ldots, N\}$ — множество ветвей параллельной программы, а $E \subseteq V \times V$ — множество информационно-логических связей между ветвями. Обозначим через d_{ij} вес ребра $(i, j) \in E$, отражающий интенсивность обменов данными между ветвями i и j при выполнении программы. В настоящей статье рассмотрено два способа задания весов ребер:

1) d_{ij} — суммарный объем данных, передаваемых между ветвями i и j за время выполнения программы ($[d_{ij}]$ = байт);

2) d_{ij} — количество переданных сообщений между ветвями i и j.

Заметим, что вес d_{ij} ребра может отражать как абсолютные, так и относительные объем или количество информационных обменов. Эти значения могут быть получены путем профилирования параллельной программы.

Отображение параллельной программы на ВС задается значениями переменных $x_{ij} \in \{0,1\}$: $x_{ij} = 1$, если ветвь $i \in V$ назначена на процессорное ядро $j \in \{1,2,\ldots,N\}$, в противном случае $x_{ij} = 0$.

Для оценки эффективности отображения программы на структуру ВС можно использовать различные показатели: время выполнения программы, энергопотребление системы и др. В данной работе в качестве показателя эффективности отображения используется время T выполнения информационных обменов. Оно определяется максимальным из времен выполнения обменов ветвями программы.

Обозначим $t(i,j,p,q)$ — суммарное время взаимодействий между ветвями $i,j \in V$, назначенными на процессорные ядра p и q соответственно. Тогда $T(X) = \max\limits_{i \in V} t_i = \max\limits_{i \in V} \left\{ \sum\limits_{j=1}^{N} \sum\limits_{p=1}^{N} \sum\limits_{q=1}^{N} x_{ip} x_{jq} t(i,j,p,q) \right\}$.

Значение функции $t(i,j,p,q)$ может быть получено согласно различным моделям дифференцированных обменов (LogP, LogGP, Hockney). Например, в случае модифицированной модели Хокни (R. Hockney) функция t принимает вид $t(i,j,p,q) = \dfrac{d_{ij}}{b(p,q)}$, где $b(p,q)$ — пропускная способность канала связи между процессорными ядрами p и q.

Сформулируем задачу оптимального отображения параллельной программы на ВС с иерархической организацией:

$$T(X) = \max_{i \in V} \left\{ \sum_{j=1}^{N} \sum_{p=1}^{N} \sum_{q=1}^{N} x_{ip} x_{jq} t(i,j,p,q) \right\} \to \min_{(x_{ij})} \tag{1}$$

при ограничениях

$$\sum_{j=1}^{N} x_{ij} = 1, \quad i = 1,2,\ldots,N, \tag{2}$$

$$\sum_{i=1}^{N} x_{ij} = 1, \quad j = 1,2,\ldots,N, \tag{3}$$

$$x_{ij} \in \{0,1\}, \quad i \in V, \quad j \in \{1,2,\ldots,N\}. \tag{4}$$

Ограничения (2) и (4) гарантируют назначение каждой ветви параллельной программы на единственную ЭМ. Ограничение (3) обеспечивает назначение на машину одной ветви. Задача (1)–(4) относится к дискретной оптимизации и является трудноразрешимой.

Рассмотрим приближенный метод ее решения.

2. Метод отображения параллельных программ на иерархические ВС. Метод основан на разбиении графа задачи на подмножества интенсивно обменивающихся параллельных ветвей и отображения их на ЭМ, связанные быстрыми каналами связи. Цель разбиения — минимизация суммы весов ребер, инцидентных разным подмножествам разбиения. Разбиение выполняется многократно для каждого уровня иерархии коммуникационной среды (рис. 2). Назовем описанный метод `HierarchicMap`. Ниже после рис. 2 приведен выделенный рамкой листинг с его псевдокодом.

Суть метода рассмотрим на примере отображения параллельной программы на подсистему из 16 ЭМ (рис. 1). На первом шаге выполняется разбиение (PartGraph) исходного графа G на n_{11} подграфов (G_{21} и G_{22}) по c_{21} и c_{22} вершин. Далее графы G_{21} и G_{22} рекурсивно разбиваются на n_{21} и n_{22} частей по c_{21} и c_{22} вершин соответственно. Полученные в результате подграфы G_{31}, G_{32}, G_{33} (их вершины — ветви программы) назначаются на узел 1 (процессорные ядра $1,\ldots,4$) и узел 2 (процессорные ядра $5,\ldots,8$) кластера A и узел 1 (процессорные ядра $9,\ldots,16$) кластера B.

Функция `PartGraph` возвращает список подграфов, получаемых в результате разбиения исходного графа. Задача оптимального разбиения взвешенного графа на k непересекающихся подмножеств является трудноразрешимой. Для ее решения существуют точные и приближенные алгоритмы различной вычислительной сложности. Интерес представляют многоуровневые алгоритмы разбиения графов [20, 21], позволяющие получать субоптимальные решения этой задачи и характеризующиеся невысокой вычисли-

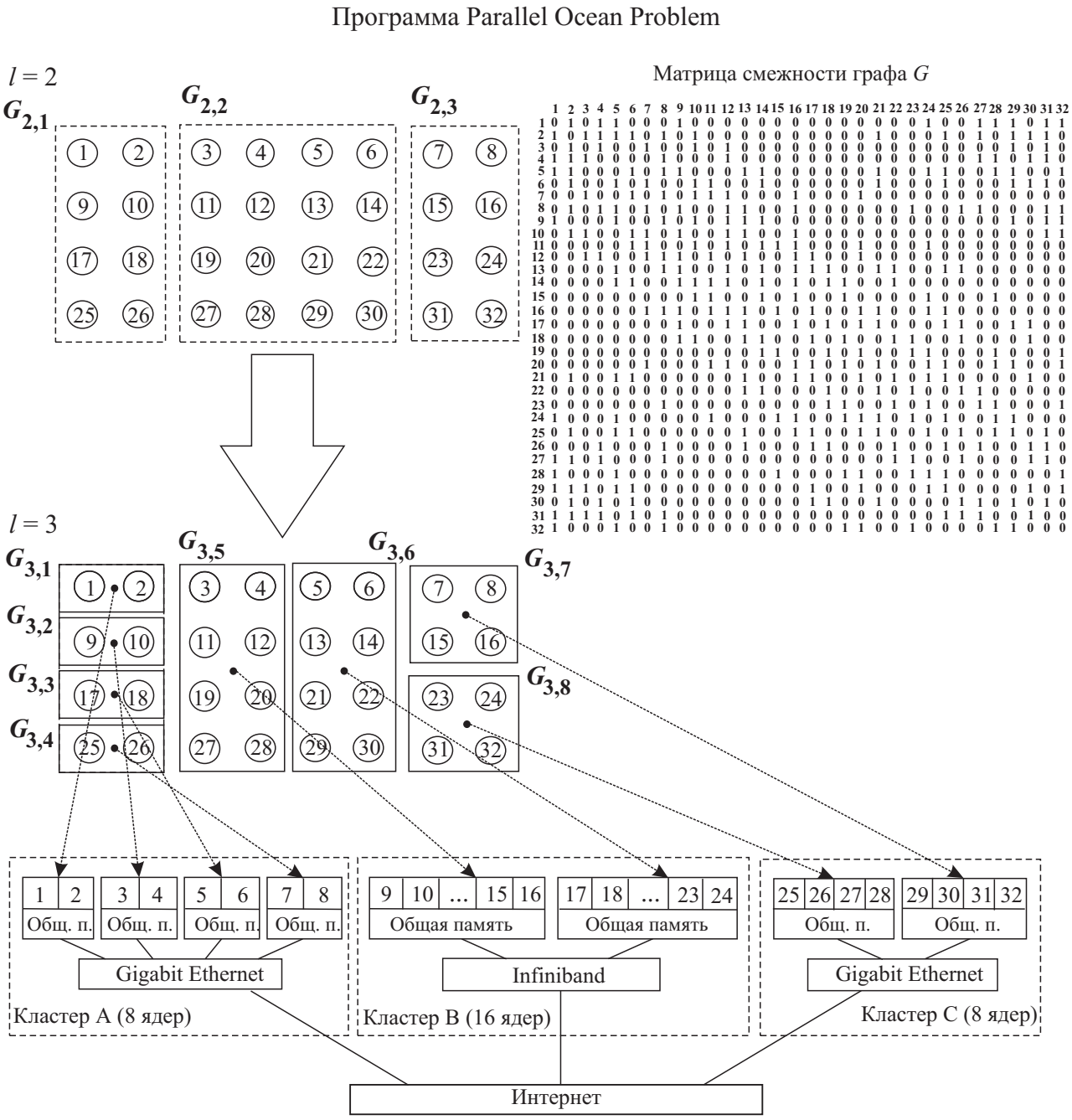

Рис. 2. Отображение программы Parallel Ocean Problem (POP) на подсистему из $N = 32$ ЭМ методом `HierarchicMap`

тельной сложностью. Данные алгоритмы включают этапы сжатия графа, начального разбиения и улучшения разбиения. В основе большинства алгоритмов улучшения разбиений лежит модифицированная нетрудоемкая эвристика Кернигана–Лина (Kernighan–Lin) [22].

Вычислительная сложность метода `HierarchicMap` определяется количеством выполнений процедуры `PartGraph`. В случае если используется многоуровневый алгоритм разбиения графа, трудоемкость ее составляет $T_{PG} = O\big(|E| \log_2 z\big)$, где $|E|$ — количество ребер в графе, z — число подмножеств разбиения графа. Разбиение выполняется в пределах каждого уровня $l = 1, \ldots, L-1$ для всех его элементов $k = 1, \ldots, n_l$ (графы G_{lk}). Время работы метода приведено на рис. 3.

На основе метода `HierarchicMap` предложены алгоритмы, различающиеся между собой уровнями l коммуникационной среды, которые учитываются при формировании разбиения. Назовем `L1Map` алгоритм, учитывающий только уровень $l = 1$ связи между подсистемами (кластерами) и не учитывающий уровень

Псевдокод метода `HierarchicMap`(G_{lk}, l, k)

Входные данные: G — информационный граф параллельной программы,
l — уровень коммуникационной среды,
k — номер текущего элемента.

Выходные данные: x_{ij} — отображение; $x_{ij} = 1$, если ветвь i назначена на ЭМ j, иначе $x_{ij} = 0$.

```
1  if l = L then
2     return G_{L,1}, G_{L,2}, ..., G_{L,n_L}
3  else
4     (G_{l+1,1}, G_{l+1,2}, ... G_{l+1,n_lk}) ← PartGraph(G_lk; n_lk; c_{l+1,1}, c_{l+1,2}, ..., c_{l+1,n_lk})
5     for k = 1 to n_lk do
6        HierarchicMap(G_{l+1,k}, l + 1, k)
7     end for
8  end if
```

связи между узлами, `L2Map` — алгоритм, учитывающий только уровень $l = 2$ связи между узлами и не учитывающий уровень связи между подсистемами, `L12Map` — алгоритм, учитывающий как уровень связи между узлами, так и уровень связи между подсистемами, и т.д.

3. Моделирование работы алгоритмов. Моделирование алгоритмов отображения проводилось на мультикластерной ВС Центра параллельных вычислительных технологий Сибирского государственного университета телекоммуникаций и информатики (ЦПВТ ФГОБУ ВПО “СибГУТИ”) и лаборатории вычислительных систем Института физики полупроводников им. А. В. Ржанова СО РАН (ИФП СО РАН).

3.1. Средства запуска параллельных программ. Для запуска MPI-программ на ресурсах пространственно-распределенных подсистем реализован подход на основе межсетевого протокола IPv6 (рис. 4). В этом протоколе используется 128-битовая адресация, которая позволят выделять всем вычислительным узлам пространственно-распределенной ВС глобальные IP-адреса. Наличие уникальных адресов обеспечивает возможность установления сетевого IP-соединения между любой парой узлов системы для запуска на них MPI-программ. Получение адресов и взаимодействие между узлами подсистем осуществлялось с помощью протокола 6to4, который реализует передачу пакетов протокола IPv6 поверх сети IPv4. При этом сначала головным узлам, имеющим глобальный IPv4-адрес, по протоколу 6to4 выделялись IPv6-адреса, после чего средствами службы radvd IPv6-адреса от головных узлов всех подсистем раздавались вычислительным узлам.

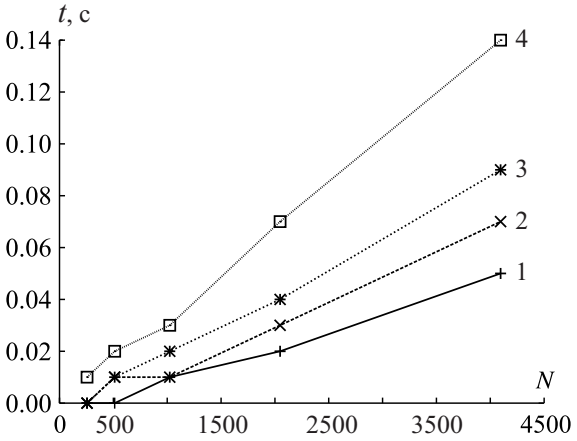

Рис. 3. Зависимость времени работы алгоритма `L1Map` от количества N ветвей в параллельной программе для различного числа k подмножеств разбиения (процессор Intel Xeon E5420): 1) $k = 16$, 2) $k = 32$, 3) $k = 64$, 4) $k = 128$

Для запуска MPI-программ пользователь должен зайти на одну из подсистем и создать файл со списком узлов мультикластерной ВС для выполнения параллельной программы. После этого он запускает MPI-программу утилитой mpiexec (mpirun). Описанная схема запуска MPI-программ требует от библиотеки MPI поддержки протокола IPv6. В данной работе применялась библиотека Open MPI 1.4.5.

3.2. Инструментарий оптимизации отображения на мультикластерные ВС. Разработанные алгоритмы реализованы в программном пакете MPIGridMap, который позволяет запускать MPI-программы на пространственно-распределенных ВС с субоптимальным распределением параллельных ветвей по элементарным машинам.

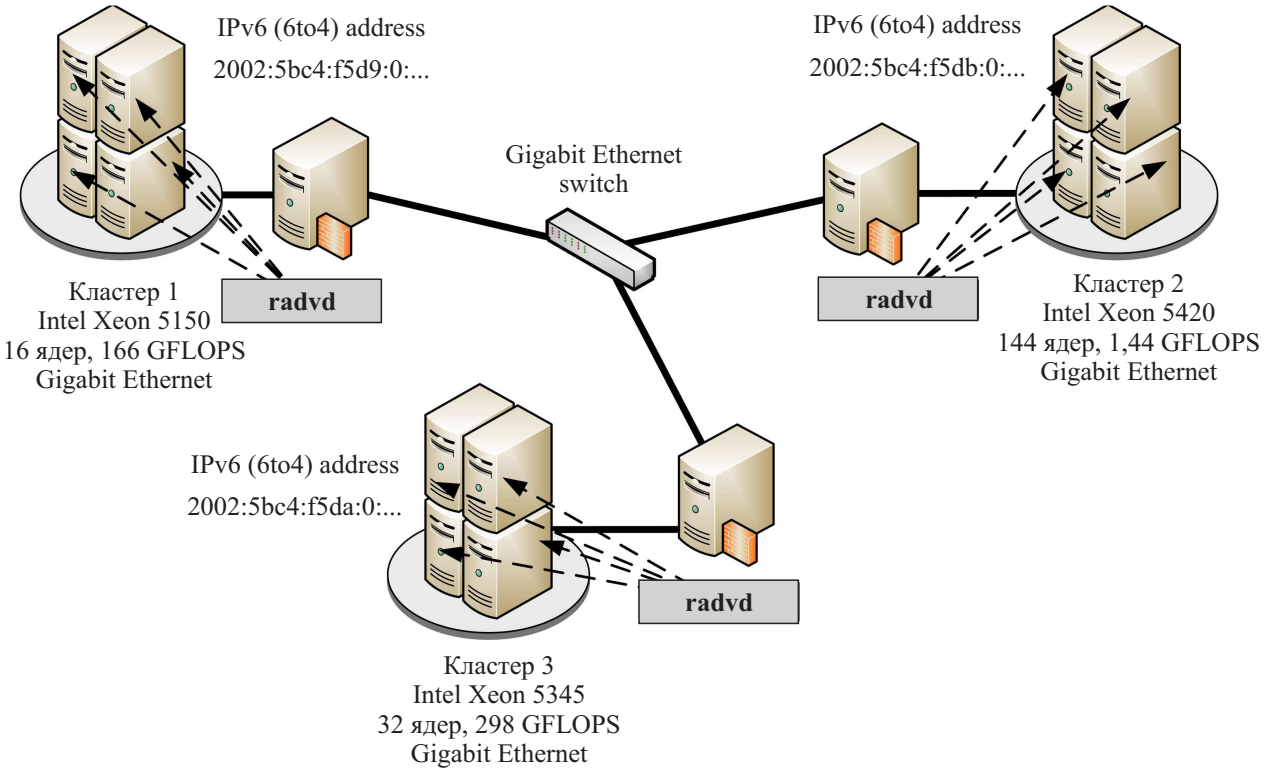

Рис. 4. Конфигурация тестовой подсистемы мультикластерной ВС

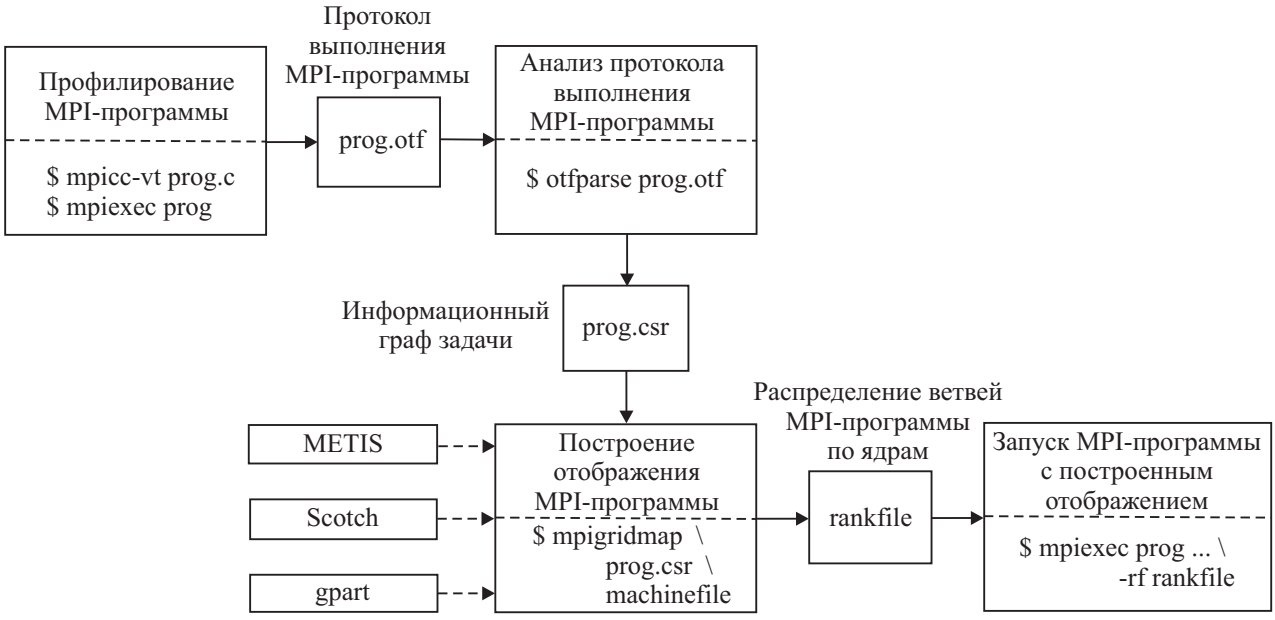

Рис. 5. Функциональная схема пакета MPIGridMap

Схема использования пакета выглядит следующим образом (рис. 5). На первом шаге программа компилируется с подключенной библиотекой профилирования VampirTrace и запускается с линейным отображением (по умолчанию). Результат профилирования — протокол `prog.otf` выполнения программы в формате Open Trace Format (OTF). Данный протокол анализируется программой OTFParse для получения информационного графа `prog.csr` параллельной задачи в формате Compressed Sparse Row (CSR). Граф отражает количество (nops) или размеры (datasize) передаваемых сообщений в дифференцированных обменах (point-to-point) при выполнении MPI-программ.

Информационный граф подается вместе со списком `machinefile` узлов мультикластерной ВС на вход

модуля MPIGridMap формирования отображения. При построении отображения могут быть использованы различные пакеты разбиения графов (METIS, Scotch, gpart). Результатом является файл `rankfile`, который содержит отображение параллельных ветвей программы на множество элементарных машин. Пользователь указывает файл `rankfile` при запуске MPI-программы.

Пакет MPIGridMap является свободно распространяемым. Он написан на языках C и C++ для операционной системы GNU/Linux.

3.3. Организация экспериментов. Натурные эксперименты по отображению тестовых MPI-программ проводились на действующей мультикластерной системе ЦПВТ ФГОБУ ВПО "СибГУТИ" и лаборатории ВС ИФП СО РАН. Тестовая подсистема (рис. 4) включала в себя три вычислительных кластера:

— сегмент D: 4 узла (2 x Intel Xeon 5150, 16 процессорных ядер),

— сегмент E: 4 узла (2 x Intel Xeon 5345, 32 процессорных ядра),

— сегмент F: 18 узлов (2 x Intel Xeon 5420, 144 процессорных ядер).

Сеть связи между узлами — Gigabit Ethernet, сеть связи между сегментами ВС — Gigabit Ethernet.

На подсистемах установлена операционная система GNU/Linux. Использовались библиотека Open MPI 1.4.5 с поддержкой IPv6 и библиотека VampirTrace для профилирования MPI-программ.

Опираясь на известные распространенные схемы межмашинных обменов [23], были выбраны следующие тестовые MPI-программы: The Parallel Ocean Program (POP, пакет моделирования климатических процессов в мировом океане), SWEEP3d (программа для моделирования процессов распространения нейтронов), GRAPH500 (тест производительности ВС на основе обработки неструктурированных данных графового типа) и программы LU, SP, MG, BT из пакета тестов производительности NAS Parallel Benchmarks.

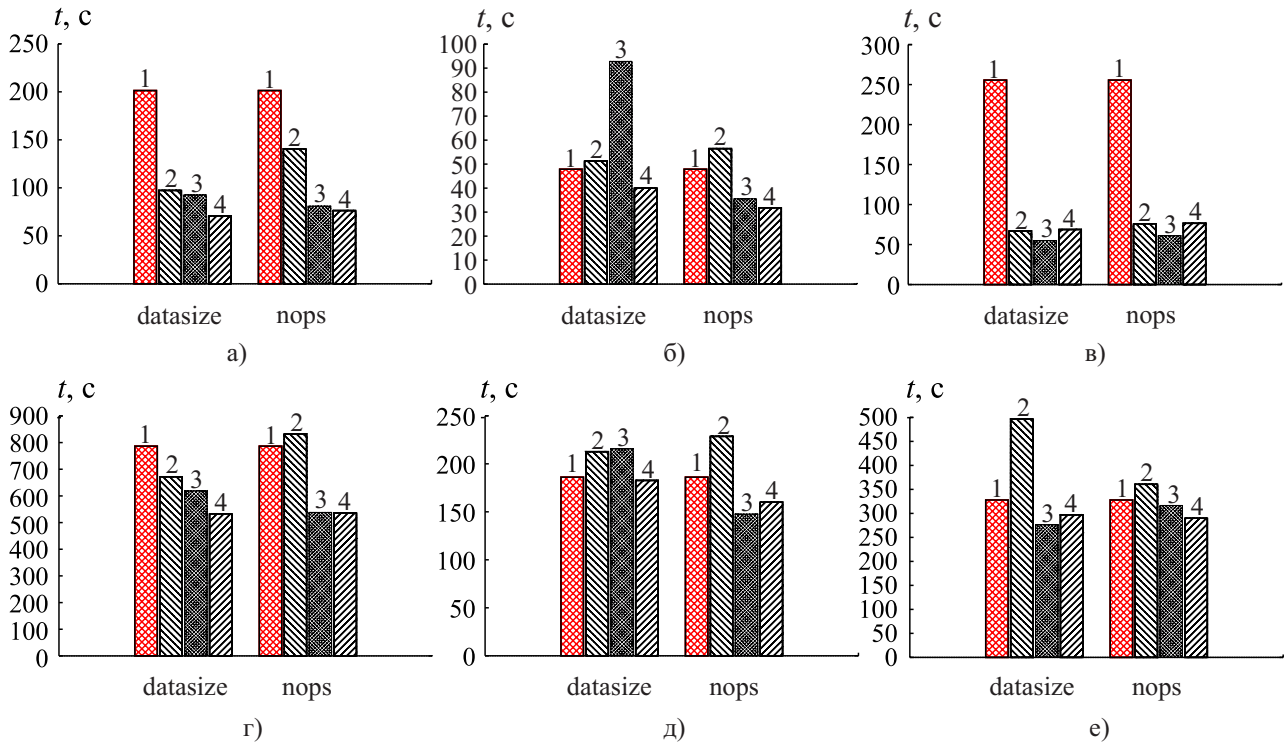

Рис. 6. Сравнение алгоритмов отображения параллельных программ: а) POP, $r = 120$, б) Sweep3D, $r = 120$, в) NPB MG, $r = 64$, г) NPB SP, $r = 64$, д) NPB LU, $r = 120$, е) NPB BT, $r = 64$; 1) линейное отображение, 2) `L2Map`, 3) `L1Map`, 4) `L12Map`

При формировании отображений использовались библиотеки разбиения графов Scotch [13], METIS [24] и gpart. Все пакеты реализуют многоуровневые алгоритмы разбиения графов с использованием прямого метода разбиения на k непересекающихся подмножеств. Следует заметить, что библиотека Scotch используется в пакете hwloc [11, 12] при отображении параллельных MPI-программ в пакетах MPICH2 и Open MPI.

Ранг r параллельной программы выбирался из множества $\{120, 64, 36, 32\}$ в зависимости от задачи.

Для каждой программы генерировалось два информационных графа в зависимости от способа формирования весов d_{ij} ребер. В первом случае вес ребра отражал объем данных, передаваемых между

ветвями i и j (datasize). Во втором графе вес ребра отражал количество информационных сообщений, передаваемых между ветвями i и j (nops).

3.4. Результаты экспериментов. На рис. 6 представлены некоторые результаты исследования алгоритмов. В процессе моделирования измерялось время выполнения параллельных программ при отображении их на мультикластерную ВС алгоритмами `L2Map`, `L1Map` и `L12Map`. Для разбиения графов задач применялся пакет gpart. Выполнялось сравнение полученных отображений с линейным отображением, при котором ветви последовательно распределяются по ЭМ выделенной подсистемы (такое отображение реализуется по умолчанию библиотеками MPI).

Время выполнения MPI-программ при отображении их алгоритмом `L12Map` меньше по сравнению с остальными алгоритмами. Это объясняется тем, что данный алгоритм позволяет учитывать как внутрикластерную сеть связи между узлами, так и сеть связи между кластерами. Различие результатов `L1Map` и `L2Map` обусловлено схемами межмашинных обменов в конкретных параллельных программах.

Уменьшение времени выполнения параллельных программ POP (в 2.5 раз, $r = 120$), Sweep3D (на 30%, $r = 120$), GRAPH500 (до 10 раз, $r = 120$), NPB MG (в 5 раз, $r = 64$), NPB SP (на 30%, $r = 64$), NPB LU (на 10%, $r = 120$) оптимизированного отображения по сравнению с линейным отображением обусловлено разреженностью и неоднородностью графов и преобладанием в программе дифференцированных MPI-обменов. В таких графах можно выделить подмножества параллельных ветвей, интенсивно обменивающиеся данными, и распределить их по ЭМ, которые соединены быстрыми каналами связи. Эффект от использования созданных алгоритмов при отображении параллельных программ с однородными графами (NPB LU, NPB BT) и с графами недетерминированных обменов незначителен.

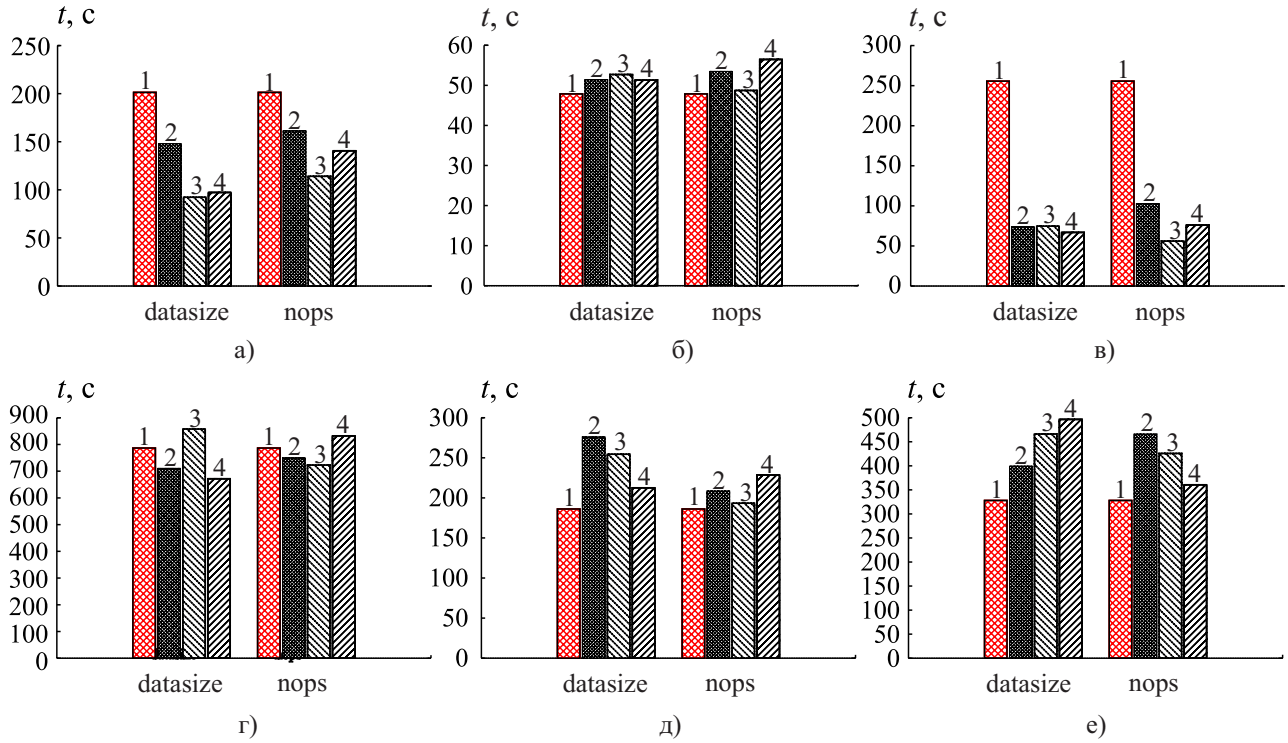

Рис. 7. Сравнение библиотек разбиения графов: а) POP, $r = 120$, б) Sweep3D, $r = 120$, в) NPB MG, $r = 64$, г) NPB SP, $r = 64$, д) NPB LU, $r = 120$, е) NPB BT, $r = 64$; 1) линейное отображение, 2) METIS, 3) Scotch, 4) gpart

Время работы алгоритмов отображения не превышает 1 с (рис. 3). Оптимальный выбор способа формирования информационного графа (datasize или nops) зависит от типа параллельной программы и от ее ранга. Если известно, что в графе параллельной программы преобладают частые обмены сообщениями небольшого размера, рекомендуется использовать тип графа nops. В остальных случаях рекомендуется использовать datasize.

Проведено моделирование с целью сравнения эффективности пакетов METIS, Scotch и gpart разбиения графов (рис. 7). В качестве алгоритма отображения применялся `L2Map`. Использование библиотеки Scotch обеспечивает незначительное сокращение времени выполнения программы POP по сравнению с другими пакетами. На остальных тестах все библиотеки дают сопоставимые или противоречивые резуль-

таты.

Заключение. Разработан метод `HierarchicMap` и алгоритмы иерархического отображения на пространственно-распределенные ВС параллельных MPI-программ с целью минимизации времени их выполнения. Суть метода заключается в последовательном разбиении графа параллельной программы в соответствии с уровнями иерархии системы, что позволяет полностью учитывать структуру коммуникационной среды пространственно-распределенных ВС. Алгоритмы могут применяться не только при отображении MPI-программ, но также в run-time системах языков семейства PGAS.

Созданные алгоритмы позволяют сократить время выполнения параллельных программ в среднем на 30%. Результаты моделирования отображения реальных MPI-программ показали, что на всех тестовых задачах алгоритм `L12Map` обеспечивает наименьшее время выполнения программ. Данный алгоритм учитывает архитектурные особенности мультикластерных и GRID-систем, в которых вычислительные узлы разных подсистем взаимодействуют через медленные каналы связи.

Эффективность отображения зависит от объема и интенсивности передаваемой информации в дифференцированных обменах. Алгоритмы позволяют существенно (до 5 раз) снизить время выполнения параллельных программ с информационными графами, имеющими разреженную структуру с преобладанием дифференцированных обменов (например, POP, NPG MG и GRAPH500).

Все предложенные алгоритмы характеризуются низкой вычислительной сложностью и позволяют отображать параллельные программы на большемасштабные ВС. Для уменьшения времени отображения алгоритмы могут быть эффективно распараллелены.

список литературы

1. *Хорошевский В.Г.* Распределенные вычислительные системы с программируемой структурой // Вестник СибГУТИ. 2010. **10**, № 2. 3–41.
2. *Gabriel E., Resch M., Rühle R.* Implementing MPI with optimized algorithms for metacomputing // Proc. of the Third MPI Developer's and User's Conference. Atlanta, 1999. 31–41.
3. *Fernandez E., Heymann E., Senar M.A.* Supporting efficient execution of MPI applications across multiple sites // Proc. of Euro-Par'2006. Berlin: Springer, 2006. 383–392.
4. *Takano R., Matsuda M., Kudoh T., Kodama Y., Okazaki F., Ishikawa Y., Yoshizawa Y.* High performance relay mechanism for MPI communication libraries run on multiple private IP address clusters // Proc. of 8th IEEE International Symposium on Cluster Computing and the Grid (CCGRID 2008). Lyon, 2008. 401–408.
5. *Saito H., Taura K.* Locality-aware connection management and rank assignment for wide-area MPI // Proc. of the 7th IEEE International Symposium on Cluster Computing and the Grid (CCGRID 2007). Rio de Janeiro, 2007. 249–256.
6. *Imamura T., Tsujita Y., Koide H., Takemiya H.* An architecture of Stampi: MPI library on a cluster of parallel computers // Proc. of the 7th European PVM/MPI'2000. Berlin: Springer, 2000. 200–207.
7. *Malyshkin N.V., Roux B., Fougere D., Malyshkin V.E.* The NumGRID metacomputing system // Bulletin of the Novosibirsk Computing Center, Computer Science Series. Issue 21. Novosibirsk, 2004. 57–68.
8. *Филамофитский М.П.* Система поддержки метакомпьютерных расчетов X-Com: архитектура и технология работы // Вычислительные методы и программирование. 2004. **5**, № 1. 123–137.
9. *Anderson D.P.* Boinc: a system for public-resource computing and storage // 5th IEEE/ACM International Workshop on Grid Computing. Washington: IEEE Press, 2004. 4–10.
10. *Хорошевский В.Г., Курносов М.Г.* Алгоритмы распределения ветвей параллельных программ по процессорным ядрам вычислительных систем // Автометрия. 2008. **44**, № 2. 56–67.
11. *Broquedis F., Clet-Ortega J., Moreaud S., Furmento N., Goglin B., Mercier G., Thibault S., Namyst R.* Hwloc: a generic framework for managing hardware affinities in HPC applications // Int. Conference on Parallel, Distributed and Network-Based Processing (PDP2010). Pisa, 2010. 180–186.
12. *Mercier G., Clet-Ortega J.* Towards an efficient process placement policy for MPI applications in multicore environments // Proc. of the 16th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface. Berlin: Springer, 2009. 104–115.
13. *Pellegrini F.* Distillating knowledge about Scotch // Combinatorial Scientific Computing, Dagstuhl Seminar Proc. Series. Dagstuhl, 2009. N 09061.
14. *Yu H., Chung I.-H., Moreira J.* Topology mapping for Blue Gene/L supercomputer // Proc. of SC'06. New York: ACM, 2006. N 116.
15. *Bhanot G.* Optimizing task layout on the Blue Gene/L supercomputer // IBM Journal of Research and Development. 2005. **49**, N 2. 489–500.
16. *Balaji P., Gupta R., Vishnu A., Beckman P.* Mapping communication layouts to network hardware characteristics on massive-scale Blue Gene systems // Special edition of the Springer Journal of Computer Science on Research and Development (presented at the International Supercomputing Conference (ISC)). 2011. **26**. 247–256.

17. *Bhatele A., Kale L.V., Kumar S.* Dynamic topology aware load balancing algorithms for molecular dynamics applications // Proc. of the 2009 ACM International Conference on Supercomputing (ICS'09). Berlin: Springer, 2009. 110–116.
18. *Hoefler T., Snir M.* Generic topology mapping strategies for large-scale parallel architectures // Proc. of the 2011 ACM International Conference on Supercomputing (ICS'11). Tucson, 2011. 75–85.
19. *Traff J.L.* Implementing the MPI process topology mechanism // Proc. of the ACM/IEEE Conference on Supercomputing. Los Alamitos, 2002. 1–14.
20. *Hendrickson B., Leland R.* A multilevel algorithm for partitioning graphs // Proc. of ACM/IEEE conference on Supercomputing. San Diego : ACM Press, 1995. 626–657.
21. *Karypis G., Kumar V.* Multilevel k-way partitioning scheme for irregular graphs // Journal of Parallel and Distributed computing. 1998. **48**. 96–129.
22. *Fiduccia C.M., Mattheyses R.M.* A linear-time heuristic for improving network partitions // Proc. of Conference on Design Automation. New York: IEEE Press, 1982. 175–181.
23. *Asanovic K. et al.* The landscape of parallel computing research: a view from Berkeley // Electrical Engineering and Computer Sciences, University of California, Berkeley. Technical Report N UCB/EECS-2006-183. Berkeley, 2006.
24. *Abou-Rjeili A., Karypis G.* Multilevel algorithms for partitioning power-law graphs // IEEE International Parallel & Distributed Processing Symposium (IPDPS). Rhodes Island: IEEE, Press, 2006. 1–17.

Поступила в редакцию
11.12.2012