Проблемы и методы организации эффективного решения параллельных задач с учетом структур распределенных вычислительных систем*

М.Г. Курносов
Институт физики полупроводников им. А.В. Ржанова СО РАН
e-mail: mkurnosov@isp.nsc.ru

Рассматриваются проблемы создания инструментария (моделей, методов и системного программного обеспечения) организации эффективного решения параллельных задач, созданных в модели передачи сообщений (message passing), на большемасштабных распределенных вычислительных системах (ВС). Предлагаются алгоритмы оптимизации коллективных схем межмашинных обменов в них. Приводятся результаты экспериментов на действующих вычислительных кластерах.

Введение. Начиная с середины 1960-х годов распределенные вычислительные системы (ВС) активно используются как инструментальные средства решения сложных задач в различных отраслях науки и техники. В архитектурном плане распределенная ВС [1 – 2] представляется композицией множества взаимодействующих элементарных машин (оснащенных средствами коммуникаций и внешними устройствами) и сети межмашинных связей. Элементарная машина (ЭМ) – это основной функциональный и структурный элемент ВС; конфигурация ЭМ допускает варьирование в широких пределах – от процессорного ядра до ЭВМ. Все основные ресурсы распределенных ВС (арифметико-логические устройства, память, средства управления и коммуникаций) являются логически и технически рассредоточенными. Число ЭМ в распределённых ВС допускает варьирование от нескольких единиц до сотен тысяч.

Параллельные алгоритмы и программы для распределенных ВС разрабатываются в модели передачи сообщений (Message Passing) или с использованием языков параллельного программирования (например, Co-array Fortran, Unified Parallel C, IBM X10, Cray Chapel, Charm++, High Performance Fortran). В обоих случаях для синхронизации выполнения ветвей параллельных задач используется механизм передачи сообщений по каналам межмашинных связей. В случае использования коммуникационных библиотек (например, стандарта MPI, PVM), пользователь явно указывает обращение к таким процедурам, а при использовании языков параллельного программирования к коммуникационным функциям обращается исполняющая подсистема.

Современные распределенные ВС являются мультиархитектурными. В зависимости от уровня рассмотрения их функциональных структур, они могут выглядеть и как MISD, и как SIMD, и как MIMD системы. Для таких систем характерна иерархическая структура и зависимость времени передачи данных между ЭМ от их размещения в системе [3 – 4].

* Работа выполнена в рамках интеграционного проекта № 113 СО РАН и при поддержке РФФИ (гранты № 08-07-00018, 09-07-13534, 09-07-90403, 09-07-12016) и Совета по грантам Президента РФ для поддержки ведущих научных школ (грант НШ-5176.2010.9).

Эффективность решения параллельных программ на распределенных ВС (в частности, по времени выполнения) во многом определяется используемыми в системах параллельного программирования подходами к решению нижеследующих проблем.

1. Использование нетрудоёмких алгоритмов организации функционирования с вычислительной сложностью и требуемой памятью порядка не выше $O(\log_2 n)$, где n – количество ЭМ в системе.

2. Обеспечение (суб)оптимального вложения в структуру распределенной ВС графа информационных обменов между ветвями параллельной программы.

3. Реализация коллективных обменов информацией между ветвями параллельной программы с учетом структуры распределенной ВС.

1. Нетрудоемкие алгоритмы организации функционирования. При организации решения параллельных задач в коммуникационных библиотеках, реализующих модель передачи сообщений, и runtime-системах языков параллельного программирования возникает необходимость хранить списки идентификаторов (адресов) ветвей параллельной программы. С увеличением количества ЭМ в распределенных ВС возникает необходимость в создании альтернативных структур данных для хранения служебной информации, как правило с вычислительной сложностью порядка не более $O(\log_2 n)$. Кроме того интерфейс библиотек и систем параллельного программирования может быть не масштабируемым и требовать модификации. Например, в стандарте MPI векторные варианты функций коллективных обменов (MPI_Alltoallw, MPI_Gatherv) требуют передачи им массивов с количеством элементов равным количество ветвей в коммуникаторе.

2. Вложение параллельных программ в структуры распределенных ВС. Время выполнения параллельных программ на распределенных ВС существенно зависит от того, насколько они эффективно вложены в систему. Под эффективным вложением (mapping) понимается такое распределение ветвей параллельной программы между ЭМ системы, при котором достигаются минимумы накладных расходов на межмашинные обмены информацией и дисбаланса загрузки ЭМ.

Задача оптимального вложения параллельной программы в распределенную ВС трудноразрешима. Актуальной является разработка нетрудоемких методов и алгоритмов вложения параллельных программ в распределенные ВС.

Применимость известных методов и алгоритмов вложения параллельных программ в современные распределенные ВС с иерархической структурой ограничена рядом причин. 1) Часть известных алгоритмов опираются на модели ВС, которые неадекватны мультиархитектурной организации современных систем. Многие описанные в литературе алгоритмы изначально ориентированы на ВС с однородной структурой сети межмашинных связей (например, в виде гиперкуба, 3D-тора или D_n-графа). 2) В ряде работ учитывается лишь структура информационного графа программы и игнорируются объемы и интенсивности обменов данными между ветвями параллельной программы. Однако, учет таких параметров практически необходим.

В коммерческих и свободно распространяемых средствах управления ресурсами распределенных ВС (TORQUE, Altair PBS Pro, SLURM, IBM LoadLeveler и др.) реализуются алгоритмы оптимизации вложения, не учитывающие структур информационных графов параллельных программ. Востребованными являются децентрализованные методы вложения

параллельных программ в большемасштабные распределенные ВС с иерархической структурой.

3. Структурно-ориентированные алгоритмы коллективных обменов. В модели передачи сообщений выделяют два типа обменов информацией между ветвями параллельных программ [1]: дифференцированный (point-to-point) и коллективные (collective) обмены. При дифференцированном обмене осуществляется передача информации из одной ветви в любую другую ветвь. Коллективные обмены подразделяются на несколько видов: трансляционный (ТО, One-to-all Broadcast), трансляционно-циклический (ТЦО, All-to-all Broadcast) и коллекторный обмены (КО, All-to-one broadcast). При трансляционном обмене данные из одной ветви передаются во все остальные; при трансляционно-циклическом обмене информация из ветвей передается каждой ветви и принимается из всех; коллекторный обмен подразумевает прием информации из всех ветвей в одну.

Анализ использования в параллельных алгоритмах и программах коллективных обменов показывает, что суммарное время их выполнения достигает 80% времени выполнения всех обменов [5]. Как правило, количество обращений в алгоритмах и программах к коллективным операциям обменов имеет функциональную зависимость от размера входных данных и в среднем находится в интервале от 10^1 до 10^4 [6].

В коммуникационных библиотеках системах параллельного программирования для реализации коллективных обменов используются алгоритмы рассылки данных по кольцу, рекурсивного сдваивания, алгоритм Дж. Брука (J. Bruck) и алгоритмы, упорядочивающие ветви в деревья различных видов [7, 8]. Перечисленные алгоритмы характеризуются различным временем выполнения и опираются на предположение об однородности каналов связи между ЭМ распределенных ВС.

В данной работе предлагается метод оптимизации алгоритмов реализации коллективных обменов, учитывающий иерархическую структуру современных распределенных ВС и производительность каналов связи между ЭМ системы. Суть подхода демонстрируется на примере создания структурно-ориентированных алгоритмов реализации трансляционно-циклических обменов.

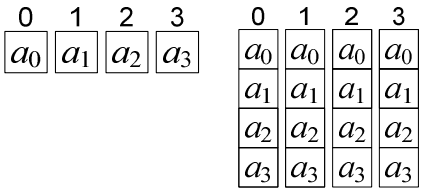
4. Алгоритмы реализации трансляционно-циклических обменов. При реализации ТЦО каждая ветвь $i \in \{0, 1, \ldots, n-1\}$ параллельной программы располагает локальным сообщением a_i размером m байт. Требуется отправить сообщение a_i всем ветвям и получить сообщения от каждой. По окончанию ТЦО все ветви должны содержать в своей памяти n сообщений $a_0, a_1, ..., a_{n-1}$, упорядоченных по номерам ветвей, которым они принадлежат (рис. 1).

Кольцевой алгоритм (Ring) реализации ТЦО организует параллельные ветви в логическое кольцо [7]. На шаге $k = 0, 1, \ldots, n-2$ ветвь i передает сообщение $a_{(i-k+n) \bmod n}$ ветви $(i+1) \bmod n$ и принимает сообщение $a_{(i-k-1) \bmod n}$ от ветви $(i-1+n) \bmod n$.

Алгоритм рекурсивного сдваивания (Recursive Doubling) допускает реализацию ТЦО только для случая n равного степени двойки. На шаге $k = 0, 1, \ldots, \log_2 n - 1$ ветви i и $i \oplus 2^k$ обмениваются ранее принятыми 2^k сообщениями (здесь и далее $\oplus$ – сложение по

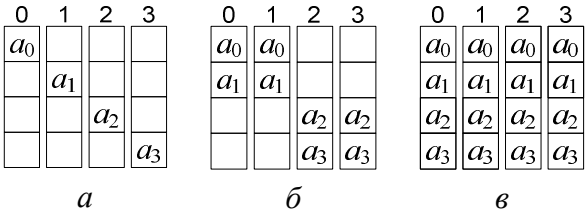
модулю 2). Таким образом, на шаге 0 передается сообщение размером m байт, на шаге 1 – размером $2m$ байт и т. д. (рис. 2).

Алгоритм Дж. Брука (J. Bruck) [8] подобен алгоритму рекурсивного сдваивания, но допускает реализацию ТЦО для произвольного количества n параллельных ветвей. На шаге $k = 0, 1, \ldots, \lceil \log_2 n \rceil - 1$ ветвь i передает все принятые сообщения ветви $(i - 2^k + n) \bmod n$ и принимает сообщения от ветви $(i + 2^k) \bmod n$. Сообщения размещаются в памяти со смещением, поэтому по окончанию работы алгоритма каждая ветвь i циклически сдвигает сообщения на i позиций вниз (рис. 3)



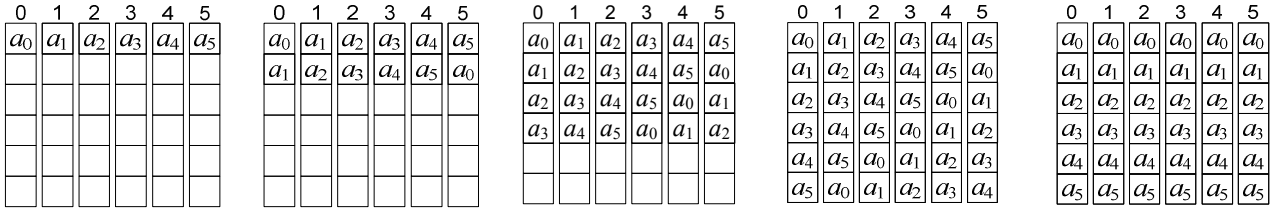
а *б*

Рис. 1. Пример реализации ТЦО ($n = 4$):
а – начальное состояние ветвей;
б – конечное состояние ветвей



а *б* *в*

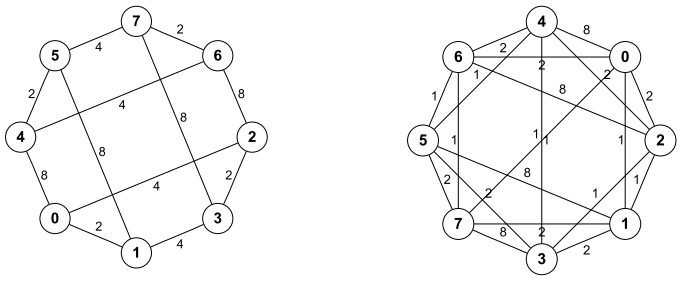
Рис. 2. Реализация ТЦО алгоритмом рекурсивного сдваивания ($n = 4$):
а – начальное состояние ветвей;
б – шаг 0; *в* – шаг 1



а *б* *в* *г* *д*

Рис. 3. Реализация ТЦО алгоритмом Дж. Брука ($n = 6$):
а – начальное состояние ветвей; *б* – шаг 0; *в* – шаг 1; *г* – шаг 2; *д* – сдвиг сообщений

Параллельная программа, реализующая ТЦО, может быть представлена информационным графом $G = (V, E)$, вершинам которого соответствуют ветви программы, а ребрам – информационные обмены между ними. На рис. 4 приведены информационные графы алгоритмов рекурсивного сдваивания и Дж. Брука. Вес d_{ij} ребра отражает объем данных переданных по нему при реализации алгоритма.



а *б*

Рис. 4. Информационные графы алгоритмов реализации ТЦО ($n = 8$; $m = 1$):
а – граф алгоритма рекурсивного сдваивания; *б* – граф алгоритма Дж. Брука

Объемы данных, передаваемых между ветвями параллельной программы при реализации ТЦО алгоритмами рекурсивного сдваивания и Дж. Брука, различны. По этой причине время реализации ТЦО рассматриваемыми алгоритмами в иерархических распределенных ВС зависит от того, как ветви параллельной программы распределены по элементарными машинам системы. Рассмотрим модель иерархической организации распределенных ВС.

5. Модель распределенных ВС с иерархической структурой. Распределенная вычислительная система с иерархической структурой, укомплектованная N однородными ЭМ, может быть представлена в виде дерева, содержащего L уровней. Каждый уровень системы образован отдельным видом функциональных модулей (например, телекоммуникационные шкафы, вычислительные узлы и т. п.), которые объединены каналами связи своего уровня. Введем следующие обозначения: $t_l(m)$ – время передачи сообщения размером m байт между парой элементарных машин ВС через каналы связи уровня $l \in \{1, 2, \ldots, L\}$; b_l – максимальное значение пропускной способности каналов связи на уровне уровня l; $z(p,q)$ – номер уровня функционального модуля, являющегося ближайшим общим предком для ЭМ $p, q \in \{1, 2, \ldots, N\}$, иначе говоря, номер уровня, через который они взаимодействуют.

Вычислительные системы, представленные в списке TOP500 (35 редакция, июнь 2010 года), имеют как минимум два уровня в иерархической организации – сеть связи между вычислительными узлами и их общая память.

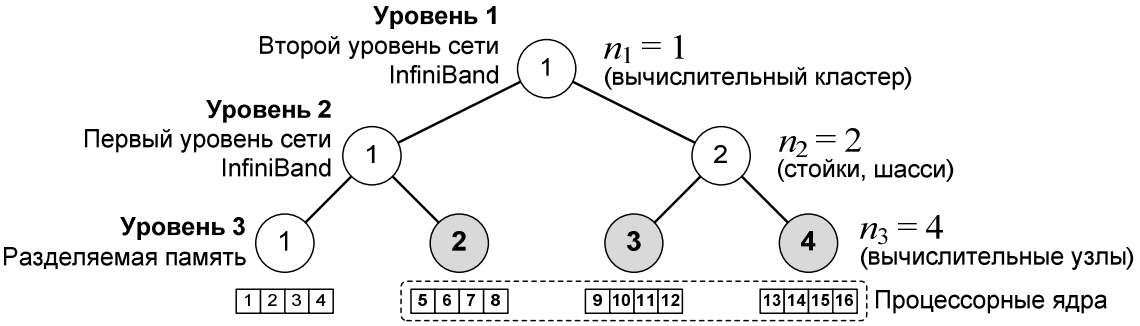


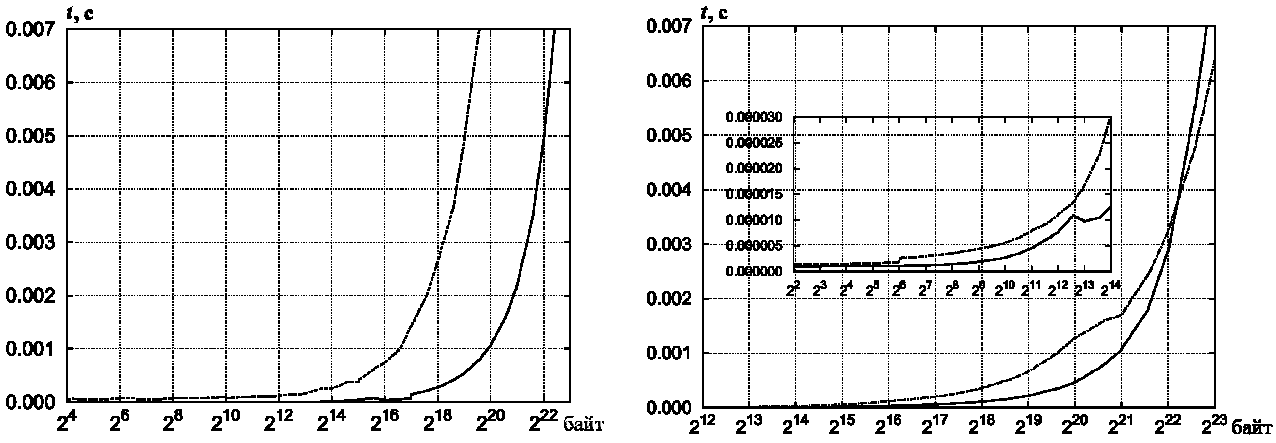
Рис. 5. Вычислительный кластер с иерархической структурой: $N = 16$, $L = 3$, $z(9,14) = 2$

На рис. 6 показано время передачи данных через каналы связи различных уровней между ЭМ кластерных ВС: кластера Центра параллельных вычислительных технологий ГОУ ВПО “Сибирский государственный университет телекоммуникаций и информатики” (ЦПВТ ГОУ ВПО “СибГУТИ”) и кластера Информационно-вычислительного центра ГОУ ВПО “Новосибирский государственный университет” (ИВЦ ГОУ ВПО “НГУ”).

Время передачи данных между ЭМ в иерархических ВС сокращается с уменьшением номера уровня, через каналы связи которого они взаимодействуют. Формально

$$\exists m_1,\ m_2 :\ \forall m \in [m_1, m_2]\ \ t_{l-1}(m) > t_l(m). \qquad (1)$$

В тоже время, в зависимости от конфигурации распределенной ВС, могут существовать интервалы размеров сообщений, для которых неравенство (1) не выполняется. Например (рис. 5*б*), для сообщений с размерами $m > 2^{22}$ байт. Рассмотрим особенности реализации ТЦО в иерархических распределенных ВС.

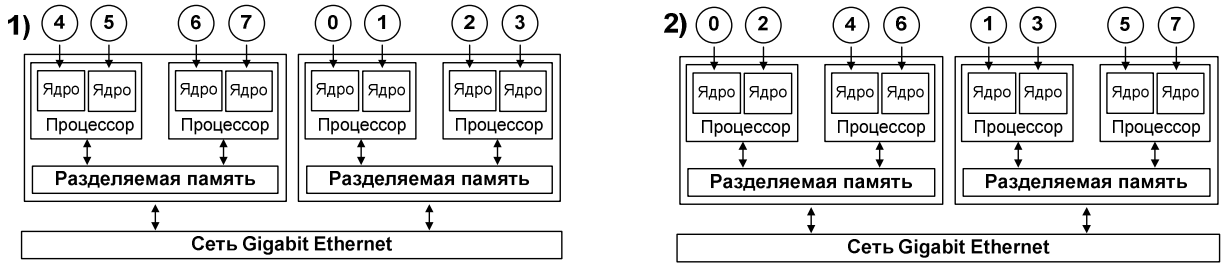


а *б*

Рис. 6. Время передачи данных между элементарными машинами кластерной ВС через общую память вычислительного узла и сеть межузловых связей:
а – кластер ЦПВТ ГОУ ВПО "СибГУТИ" *б* – кластер ИВЦ ГОУ ВПО "НГУ"
—— – разделяемая память; ---- – сеть Gigabit Ethernet; –·–·– – сеть InfiniBand 4x DDR

6. Реализации ТЦО в иерархических распределенных ВС. Обозначим через $x_i \in \{1, 2, \ldots, N\}$ номер ЭМ, на которую распределена ветвь $i \in \{0, 1, \ldots, n-1\}$ программы; h – количество функциональных модулей уровня L (вычислительных узлов), выделенных для реализации программы; $q_1, q_2, ..., q_h$ – номера выделенных функциональных модулей; s_r – количество ЭМ функционального модуля q_r, выделенных для реализации программы, $r \in \{1, 2, \ldots, h\}$. Пример подсистемы ЭМ с параметрами $h = 3$, $q_1 = 2$, $q_2 = 3$, $q_3 = 4$, $q_1 = 2$, $s_1 = s_2 = s_3 = 4$ показан на рис. 5.

На рис. 7 приведен пример реализации ТЦО алгоритмом Дж. Брука на вычислительном кластере с иерархической организацией. Время выполнения алгоритма с распределением ветвей по ЭМ стандартными средствами библиотеки MPI (MPICH2 1.2.1) составило 567 мкс, а время реализации с учетом графа алгоритма и иерархической организации кластерной ВС – 324 мкс.



а *б*

Рис. 7. Реализация ТЦО алгоритмом Дж. Брука на кластерной ВС ($n = 8$; $m = 2048$; $L = 2$):
а – реализация ТЦО с распределение ветвей стандартным алгоритмом библиотеки MPI;
б – реализация ТЦО с учетом иерархической организации ВС

Причиной сокращения времени выполнения ТЦО в 1,75 раз (рис. 7*б*) является распределение ветвей, обменивающихся большими объемами данных, на один вычислительный узел, в рамках которого они взаимодействуют через его общую память. Учитывая последнее, разработаны метод и алгоритмы реализации ТЦО в иерархических распределенных ВС.

Накладные расходы на реализацию ТЦО с заданным распределением $X = (x_0, x_1, ..., x_{n-1})$ параллельных ветвей по ЭМ системы и заданным информационным графом $G = (V, E)$ алгоритма могут быть выражены как сумма $T(X)$ времен передачи данных между ветвями программы

$$T(X) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} d_{ij} \Big/ b_{z(x_i, x_j)},$$

где d_{ij} – вес ребра в графе алгоритма, а $b_{z(x_i, x_j)}$ – пропускная способность каналов связи на уровне $z(x_i, x_j)$, через которые взаимодействую ветви i и j.

В основе разработанного метода лежит процедура распределения параллельных ветвей программы, обменивающихся большими объемами данных при реализации ТЦО заданным алгоритмом, на ЭМ одного функционального модуля. Это обеспечивает локализацию взаимодействий ветвей через каналы связи функционального модуля и, как следствие, сокращение времени информационных обменов. Например, при реализации ТЦО алгоритмом Дж. Брука (рис. 4*б*) суммарный объем данных, передаваемых между ветвями 0, 2, 4 и 6, больше объема данных, отправляемых другим ветвям, поэтому распределение этих ветвей на один вычислительный узел (рис. 7*б*) обеспечило их взаимодействие через общую память узла и сокращение времени реализации ТЦО.

Метод подразумевает (суб)оптимальное распределение ветвей по ЭМ и реализацию ТЦО в соответствии с ним. Рассмотрим основные шаги метода, осуществляемые до реализации ТЦО.

1. Формируется взвешенный информационный граф $G = (V, E)$ алгоритма реализации ТЦО для $m = 1$ с количеством вершин n.

2. Строится разбиение $R = (r_0, r_1, ..., r_{n-1})$ информационного графа $G = (V, E)$ на h непересекающихся подмножеств $V_1, V_2, ..., V_h$ с целью минимизации суммарного веса рёбер, инцидентных различным подмножества разбиения. Через $r_i \in \{1, 2, \ldots, h\}$ обозначен номер подмножества разбиения, к которому отнесена вершина $i \in V$. Количество элементов в подмножестве V_u должно быть равно заданному числу s_u, $u = 1, 2, \ldots, h$. Параметры h и s_u определены ранее.

Обозначим множество ребер инцидентных различным подмножествам разбиения через

$$E' = \{(i, j) \in E \mid r_i \neq r_j, i \neq j\}.$$

Тогда вес ребра $(i, j) \in E$, инцидентного вершинам из разных подмножеств $i \in V_u$ и $j \in V_v$, есть

$$d_{ij} \Big/ b_{z(x_i, x_j)}.$$

Разбиение R должно доставлять минимум целевой функции (2) и удовлетворять системе ограничений (3) – (5).

$$F(r_0, r_1, ..., r_{n-1}) = \sum_{(i, j) \in E'} d_{ij} \Big/ b_{z(x_i, x_j)} \to \min_{(r_i)} \qquad (2)$$

при ограничениях:

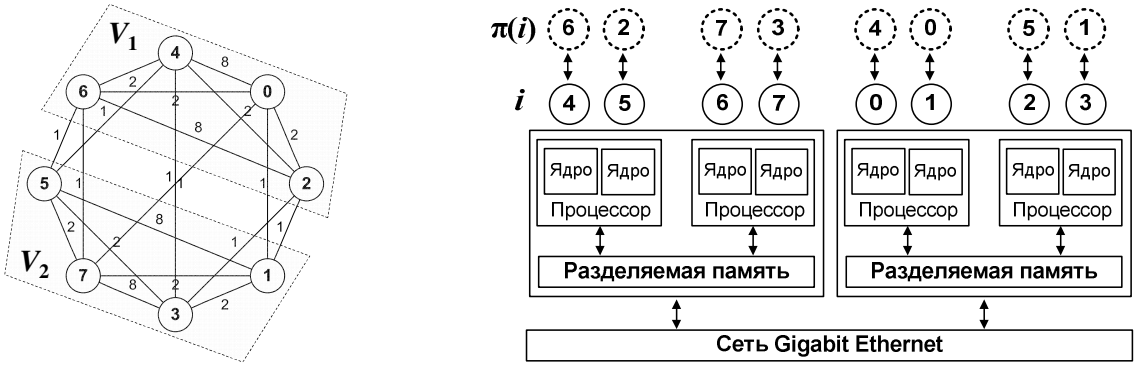
$$V_1 \cap V_2 \cap ... \cap V_h = \varnothing, \quad (3)$$

$$V_1 \cup V_2 \cup ... \cup V_h = V, \quad (4)$$

$$|V_u| = s_u, \; u = 1, 2, \ldots, h. \quad (5)$$

3. Ветвям с номерами из подмножества V_u назначаются номера ветвей, распределенных на элементарные машины функционального модуля q_u, $u = 1, 2, \ldots, h$. Таким образом, строится взаимно однозначное отображение $\pi(i)$, которое сопоставляет исходным номерам $0, 1, \ldots, i, \ldots, n-1$ ветвей новые номера $\pi(0), \pi(1), \ldots, \pi(n-1)$. Через $\pi^{-1}(i)$ обозначим отображение обратное к $\pi(i)$.

На рис. 8 показано применение описанного метода для алгоритма Дж. Брука на кластерной ВС с иерархической структурой.



а *б*

Рис. 8. Пример оптимизации алгоритма Дж. Брука ($L = 2$; $n = 8$; $h = 2$; $s_1 = 4$; $s_2 = 4$):
а – приближенное разбиение графа алгоритма Дж. Брука на подмножества V_1 и V_2;
б – отображение $\pi(i)$ номеров ветвей программы

Алгоритмы рекурсивного сдваивания и Дж. Брука требуют модификации для реализации ТЦО с заданным распределением (отображениями $\pi(i)$ и $\pi^{-1}(i)$) ветвей по ЭМ. В исходных алгоритмах каждая ветвь i располагает сообщением a_i, но в соответствии с полученным распределением ветвь должна осуществлять ТЦО под новым номером $\pi(i)$. Предложим версии алгоритмов рекурсивного сдваивания и Дж. Брука, обеспечивающие реализацию ТЦО с заданными распределениями ветвей по ЭМ.

Алгоритм Bruck exch. Ветвь i передает свое сообщение a_i ветви $\pi^{-1}(i)$, принимает от ветви $\pi(i)$ сообщение и делает его начальным. На шаге $k = 0, 1, \ldots, \lceil \log_2 n \rceil - 1$ ветвь i передает все принятые сообщения ветви $\pi^{-1}((i' - 2^k + n) \bmod n)$ и принимает сообщения от ветви $\pi^{-1}((i' + 2^k) \bmod n)$, где $i' = \pi(i)$. Далее каждая ветвь циклически сдвигает сообщения вниз на i' позиций.

Алгоритм Bruck reorder. На шаге $k = 0, 1, \ldots, \lceil \log_2 n \rceil - 1$ ветвь i передает все принятые сообщения ветви $\pi^{-1}((i' - 2^k + n) \bmod n)$ и принимает сообщения от ветви $\pi^{-1}((i' + 2^k) \bmod n)$, где $i' = \pi(i)$. Далее каждая ветвь переставляет сообщение из позиции $j = 0, 1, \ldots, n-1$ в позицию $\pi^{-1}((j + i') \bmod n)$.

Алгоритм recursive doubling exch. Ветвь i передает свое сообщение a_i ветви $\pi^{-1}(i)$, принимает от ветви $\pi(i)$ сообщение и делает его начальным. На шаге $k = 0, 1, \ldots, \log_2 n - 1$ ветвь i и $\pi^{-1}(i \oplus 2^k)$ обмениваются принятыми 2^k сообщениями.

Алгоритм recursive doubling reorder. Ветвь i сдвигает свое сообщение a_i из позиции i в позицию $\pi(i)$. На шаге $k = 0, 1, \ldots, \log_2 n - 1$ ветвь i и $\pi^{-1}(i \oplus 2^k)$ обмениваются ранее принятыми 2^k сообщениями. Далее каждая ветвь переставляет сообщение из позиции $j = 0, 1, \ldots, n-1$ в позицию $\pi^{-1}(j)$.

Формирование, разбиение графа и построение отображений $\pi(i)$ и $\pi^{-1}(i)$ осуществляется единожды при создании подсистемы ЭМ (например, при создании коммуникаторов в библиотеках стандарта MPI). После чего, построенные отображения многократно используются для реализации ТЦО. Все шаги метода эффективно реализуемы в параллельном виде на подсистеме ЭМ, выделенной для реализации программы. Также допустима организация распределенного хранения информационного графа на ЭМ системы. При создании отображений для алгоритмов рекурсивного сдваивания и Дж. Брука каждой ветви достаточно хранить списки смежных с ними вершин, количество которых не превосходит величины $2\lceil \log_2 n \rceil$. После построения отображений информационный граф алгоритма удаляется из памяти. На протяжении всего времени выполнения программы ветвям требуется лишь информация об отображении смежных с ними ветвей. Последнее, для алгоритмов рекурсивного сдваивания и Дж. Брука, требует хранения каждой ветвью в памяти порядка $2\lceil \log_2 n \rceil$ байт. Сказанное выше обеспечивает применимость метода для большемасштабных распределенных ВС.

7. Библиотека коммуникационных функций ТороMPI. Разработанный метод реализован в коммуникационной библиотеке ТороMPI, созданной и развиваемой ЦПВТ ГОУ ВПО “СибГУТИ” совместно с Лабораторией вычислительных систем Института физики полупроводников им. А.В. Ржанова СО РАН.

В библиотеке используется интерфейс профилирования стандарта MPI для перехвата обращений к функциям коллективных операций информационных обменов, остальные вызовы передаются системной библиотеке MPI (MPICH2, OpenMPI и др.). Описание иерархической организации распределенной ВС загружается библиотекой из конфигурационного файла при инициализации MPI-программы. На основе информации о распределении ветвей по ЭМ системы определяется количество h выделенных программе функциональных модулей низшего уровня (вычислительных узлов). Построение отображений номеров ветвей осуществляется при создании MPI-коммуникаторов (включая MPI_COMM_WORLD). Выбор алгоритма реализации ТЦО (функции MPI_Allgather) осуществляется на основе значения переменной среды окружения. Информационные графы алгоритмов формируются в виде списков смежных вершин в упакованном формате CSR – Compressed Sparse Row Graph. Вычислительная сложность формирования графов алгоритмов рекурсивного сдваивания и Дж. Брука составляют $O(n \log_2 n)$ и $O(n\lceil \log_2 n \rceil)$ соответственно. Разбиение графа на h подмножеств осуществляется многоуровневым алгоритмом рекурсивной бисекции [9 – 11]. Для разделения графа на два подмножества применяется метод Levelized Nested Dissection [10], основанный на обходе графа в ширину. Улучшение

разделений осуществляется алгоритмом локальной оптимизации Fiduccia-Mattheyses [10]. Вычислительная сложность алгоритма разбиения графов составляет $O((|E|+n)\log_2 h)$.

Формирования отображений номеров ветвей из подмножеств разбиения в номера ветвей, распределенных по функциональным модулям низшего уровня (вычислительным узлам) требует выполнения порядка $O(n)$ операций. Таким образом, вычислительная сложность формирования, разбиения информационного графа и создания отображений составляет $O(n\log_2 n+(|E|+n)\log_2 h)$.

8. Экспериментальное исследование алгоритмов. Исследование созданных алгоритмов проводилось на кластерных ВС ЦПВТ ГОУ ВПО "СибГУТИ" и ИВЦ ГОУ ВПО "НГУ". Кластер ЦПВТ ГОУ ВПО "СибГУТИ" построен на базе 10 двухпроцессорных узлов Intel SR2520SAFR с четырехъядерными процессорами Intel Xeon E5420. Первый уровень коммуникационной среды кластера – сеть связи стандарта Gigabit Ethernet, второй уровень – общая память вычислительных узлов. Кластер ИВЦ ГОУ ВПО "НГУ" укомплектован 64 двухпроцессорными узлами Hewlett-Packard BL460c с четырехядерными процессорами Intel Xeon 5355. Первый уровень коммуникационной среды кластера – сеть связи стандарта InfiniBand 4x DDR, второй уровень – общая память вычислительных узлов.

На кластере ЦПВТ ГОУ ВПО "СибГУТИ" пакет ТороMPI компилировался с библиотекой стандарта MPI – MPICH2 1.2.1 в операционной системе CentOS 5.2 x86-64, а на кластере ИВЦ ГОУ ВПО "НГУ" в операционной системе SUSE Linux Enterprise Server 10 SP1 x86-64 с библиотекой OpenMPI 1.2.5.

На рис. 9 показано время формирования, разбиений информационных графов алгоритмов и создание отображений $\pi(i)$, $\pi^{-1}(i)$ на процессоре Intel Xeon E5420. Видно, что время, затрачиваемое на формирование отображений номеров ветвей, незначительно. Время же разбиения информационного графа алгоритма Дж. Брука с количеством вершин $n=1048576$ на $h=131072$ подмножеств составляет 32.73 с. Поэтому для большемасштабных ВС востребованы параллельные алгоритмы формирования и разбиения графов и создания отображений.

На рис. 10 представлено среднее время ТЦО "короткими" сообщениями на подсистеме из 64 процессорных ядер (4 вычислительных узла). В качестве времени реализации ТЦО бралось максимальное из времен выполнения ветвей параллельной программы. Среднее время рассчитывалось по результатам 100 запусков алгоритма для каждого размера сообщения. Среднее время построения отображений $\pi(i)$ и $\pi^{-1}(i)$ для примера на рис. 10 составило 480 мкс и 300 мкс, соответственно, для алгоритмов Bruck exch., Bruck reorder и recursive doubling exch., recursive doubling reorder. Видно, что накладные расходы на построение отображений не значительны и компенсируются сокращением времени реализации ТЦО созданными алгоритмами.

На рис. 11 приведены значения коэффициентов ускорения созданных алгоритмов при передаче коротких и длинных сообщений относительно алгоритмов рекурсивного сдваивания и Дж. Брука. Ускорение в 130 … 180 раз на коротких сообщениях объясняется тем, что при реализации ТЦО сообщениями размером 1 Кбайт осуществляется обмен блоками 1, 2, 4, 8, 16 и 32 Кбайт, время передачи которых через сеть связи Gigabit Ethernet и общую память узла отличается в 10 … 55 раз (рис. 6*а*).

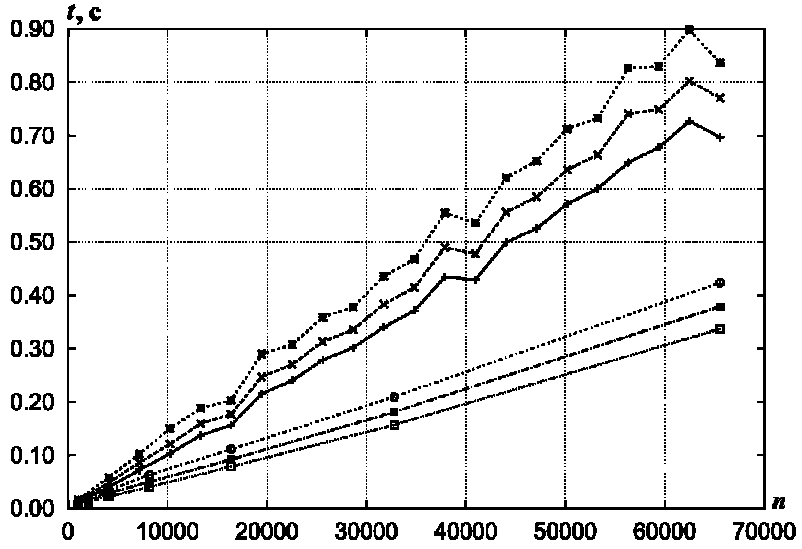


Рис. 9. Зависимость времени разбиения информационных графов на h подмножеств от количества n вершин:

—+— – граф алгоритма Дж. Брука, $h = 128$;

----×---- – граф алгоритма Дж. Брука, $h = 256$;

·····✱····· – граф алгоритма Дж. Брука, $h = 512$;

·····□····· – граф алгоритма рекурсивного сдваивания, $h = 128$;

-·-■-·- – граф алгоритма рекурсивного сдваивания, $h = 256$;

·····○····· – граф алгоритма рекурсивного сдваивания, $h = 512$

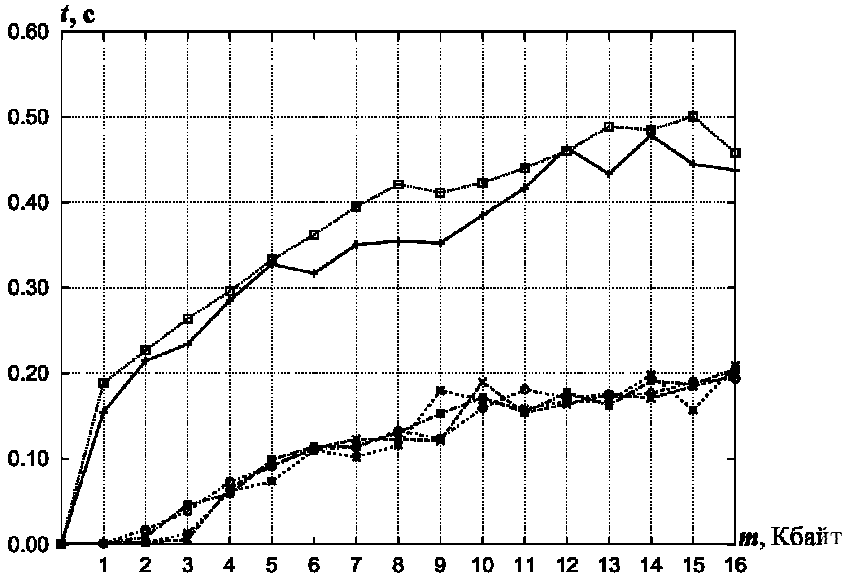


Рис. 10. Зависимость времени ТЦО от размера m сообщения на кластерной ВС ЦПВТ ГОУ ВПО “СибГУТИ” ($n = 64$; $h = 8$; $s_1 = s_2 = ... = s_8 = 8$):

—+— – алгоритм Дж. Брука;

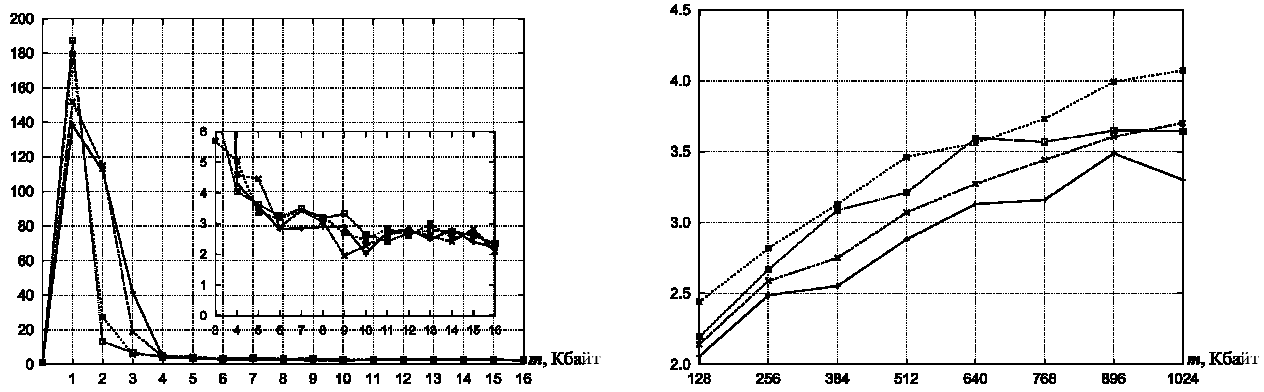
----×---- – алгоритм Bruck exch.;

·····✱····· – алгоритм Bruck reorder;

·····□····· – алгоритм рекурсивного сдваивания;

-·-■-·- – алгоритм recursive doubling exch.;

·····○····· – алгоритма recursive doubling reorder



а *б*

Рис. 11. Зависимость коэффициента ускорения алгоритмов ТЦО от размера m передаваемого сообщения на кластерной ВС ЦПВТ ГОУ ВПО “СибГУТИ” ($n = 64$; $h = 8$; $s_1 = s_2 = ... = s_8 = 8$):

а – ускорение алгоритмов на коротких сообщениях;

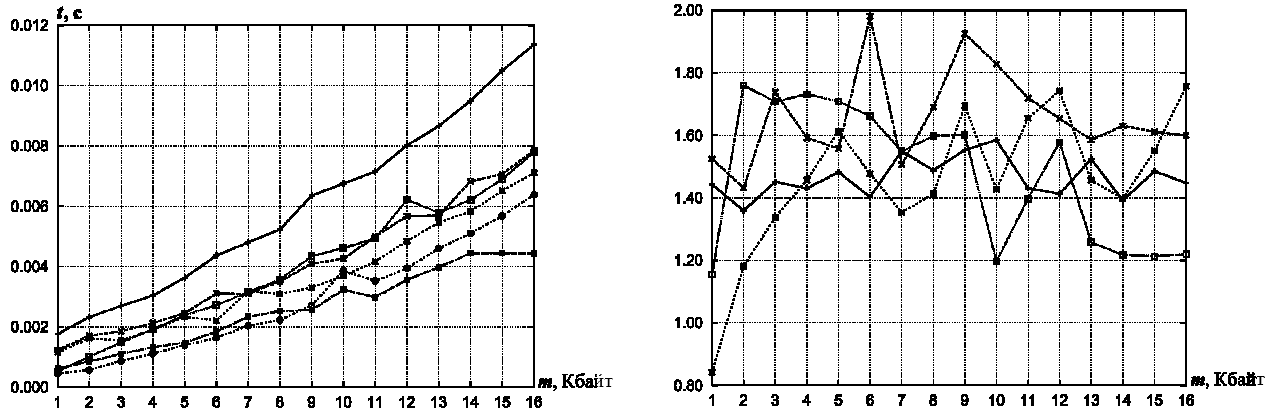
б – ускорение алгоритмов на длинных сообщениях;

——+—— – алгоритм Bruck exch.; ----×---- – алгоритм Bruck reorder;

·····✱····· – алгоритм recursive doubling exch.; ·····▣····· – алгоритма recursive doubling reorder;

Ускорение алгоритмов реализации ТЦО между 64 параллельными ветвями на кластерной ВС ИВЦ ГОУ ВПО “НГУ” (рис. 12) наблюдается только для сообщений размеров 256 байт – 128 Кбайт. Это объясняется тем, что при передаче сообщений размеров больше 2^{22} байт сеть InfiniBand (рис. 6*б*) превосходит общую память узла по времени передачи данных.

Созданные алгоритмы целесообразно применять для реализации ТЦО сообщений таких размеров, что время их передачи через каналы связи вышележащих уровней превосходит время обменов через каналы связи нижележащих уровней.



а *б*

Рис. 12. Ускорение алгоритмов на кластерной ВС ИВЦ ГОУ ВПО “НГУ”
($n = 64$; $h = 8$; $s_1 = s_2 = ... = s_8 = 8$):

а – время ТЦО короткими сообщениями;

б – коэффициент ускорение алгоритмов на коротких сообщениях

——+—— – алгоритм Дж. Брука; ----×---- – алгоритм Bruck exch.;

·····✱····· – алгоритм Bruck reorder; ·····▣····· – алгоритм рекурсивного сдваивания;

–·–■–·– – алгоритм recursive doubling exch.; ·····⊙····· – алгоритма recursive doubling reorder

Заключение. При организации эффективного решения параллельных задача на распределенных ВС необходимо учитывать их структурные характеристики. Модели и методы организации функционирования распределенных ВС должны характеризовать невысокой вычислительной сложностью и требуемым объемом памяти (порядка не выше $O(\log_2 n)$, где n – количество ЭМ в системе).

Предложенный метод оптимизации алгоритмов коллективных обменов информацией позволяет организовать межмашинные обмены с учетом иерархической организации современных распределенных ВС. Созданные версии алгоритмов рекурсивного сдваивания (Recursive Doubling) и Дж. Брука (J. Bruck) обеспечивают реализацию трансляционно-циклических обменов (ТЦО) с заданным распределением ветвей по элементарным машинам распределенной ВС. Эксперименты на действующих вычислительных кластерах показали превосходство в среднем в 1.1 – 4 раза во времени реализации ТЦО разработанными алгоритмами рекурсивного сдваивания и Дж. Брука по сравнению с исходными алгоритмами. Накладные расходы на оптимизацию незначительны и компенсируются сокращением времени при многократном обращении к функциям реализации ТЦО. Предложенный метод допускает эффективную параллельную реализацию и не предъявляет существенных требований к объему оперативной памяти, что обеспечивает возможность его применения в большемасштабных распределенных вычислительных системах.

ЛИТЕРАТУРА

1. Хорошевский В.Г. Архитектура вычислительных систем. – М.: МГТУ им. Н.Э. Баумана, 2008. – 520 с.
2. Евреинов Э.В., Хорошевский В.Г. Однородные вычислительные системы. – Новосибирск: Наука, 1978. – 320 с.
3. Balaji P., Buntinas D., Goodell D., Gropp W., Kumar S., Lusk E., Thakur R. and Traff J. L. MPI on a Million Processors // Proc. of the PVM/MPI – Berlin: Springer-Verlag, 2009. – P. 20-30.
4. Khoroshevsky V., Kurnosov M. Mapping Parallel Programs into Hierarchical Distributed Computer Systems // Proc. of “Software and Data Technologies”. – Sofia: INSTICC, 2009. – Vol. 2. – P. 123-128.
5. Rabenseifner R.. Automatic MPI Counter Profiling // Proceedings of the 42nd Cray User Group. – Noorwijk, The Netherlands, 2000. – 19 pp.
6. Han D., Jones T.. MPI Profiling // Technical Report UCRL-MI-209658 – Lawrence Livermore National Laboratory, USA, 2004. – 15 pp.
7. Thakur R., Rabenseifner R., and Gropp W. Optimization of collective communication operations in MPICH // Int. Journal of High Performance Computing Applications. – 2005. – Vol. 19, No. 1. – P. 49-66.
8. Bruck J., Ho C.-T., Kipnis S., Upfal E. and Weathersby D. Efficient algorithms for all-to-all communications in multiport message-passing systems // IEEE Transactions on Parallel and Distributed Systems. – 1997. – Vol. 8 (11). – P. 1143-1156.
9. Karypis G. and Kumar V. A Fast and Highly Quality Multilevel Scheme for Partitioning Irregular Graphs // SIAM Journal on Scientific Computing. – 1999. – Vol. 20, No. 1, P. 359-392.

10. Schloegel K., G. Karypis, V. Kumar. Graph partitioning for high-performance scientific simulations // Sourcebook of parallel computing. – San Franciso: Morgan Kaufmann Publish, 2003. – P. 491-541.
11. Курносов М.Г. Алгоритмы вложения параллельных программ в иерархические распределённые вычислительные системы // Вестник СибГУТИ. – 2009. – № 2 (6). – С. 20-45.