Моделирование алгоритмов коллективных обменов информацией между ветвями параллельных программ*

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

Предлагается метод оптимизации алгоритмов реализации коллективных обменов информацией между ветвями параллельных программ в иерархических распределенных вычислительных системах (ВС). Метод поясняется на примере создания алгоритмов трансляционно-циклических обменов (ТЦО, All-to-all Broadcast), учитывающих иерархическую структуру распределенных ВС. Приводятся результаты моделированя разработанных алгоритмов ТЦО на вычислительных кластерах различных конфигураций. Накладные расходы на оптимизацию алгоритмов незначительны и компенсируются сокращением времени реализации ТЦО предложенными алгоритмами.

1. Введение

Основу современного высокопроизводительного инструментария моделирования природных явлений и сложных технических процессов составляют распределенные вычислительные системы (ВС). Параллельные алгоритмы и программы для таких ВС преимущественно разрабатываются в модели передачи сообщений (Message Passing), в частности с использованием библиотек стандарта MPI – Message Passing Interface. В рамках этой модели пользователю доступы два типа коммуникационных операции для обмена информацией между ветвями параллельных программ: дифференцированный (Point-to-Point) и коллективные (Collective) обмены. При дифференцированном обмене осуществляется передача информации из одной ветви в любую другую ветвь. На основе таких обменов реализуются коллективные операции, которые делятся на несколько видов [1]: трансляционно-циклический (ТЦО, All-to-All Broadcast), трансляционный (ТО, One-to-All Broadcast) и коллекторный обмены (КО, All-to-One Broadcast). При трансляционно-циклическом обмене информация из ветвей передается каждой ветви и принимается из всех, при трансляционном обмене данные из одной ветви передаются во все остальные, коллекторный обмен подразумевает прием информации из всех ветвей в одной.

Анализ использования в параллельных алгоритмах и программах коллективных операций обменов показывает, что суммарное время их выполнения достигает 10 - 45 процентов времени выполнения программ [2].

В коммуникационных библиотеках стандарта MPI и системах параллельного программирования (в частности в модели PGAS – Partitioned Global Address Space) для реализации коллективных обменов используются алгоритмы рассылки данных по кольцу (Ring), рекурсивного сдваивания (Recursive Doubling), алгоритм Дж. Брука (J. Bruck) и

*Работа выполнена при поддержке РФФИ (грант №10-07-00157) и Совета по грантам Президента РФ для поддержки ведущих научных школ (грант НШ-5176.2010.9).

алгоритмы, упорядочивающие ветви в деревья различных видов [3, 4]. Перечисленные алгоритмы характеризуются различным временем выполнения и опираются на предположение об однородности каналов связи между элементарными машинами (ЭМ) распределенных ВС. Однако современные системы являются мультиархитектурными, для них характерны иерархическая структура и зависимость времени передачи данных между ЭМ от их размещения в системе [1, 4, 5].

В данной работе предлагается метод оптимизации алгоритмов 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.

2. Алгоритмы 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 $a_0, a_1, \ldots, a_n-1$, упорядоченных по номерам ветвей, которым они принадлежат. В стандарте MPI ТЦО реализуется коммуникационной функцией `MPI_Allgather`.

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 [3].

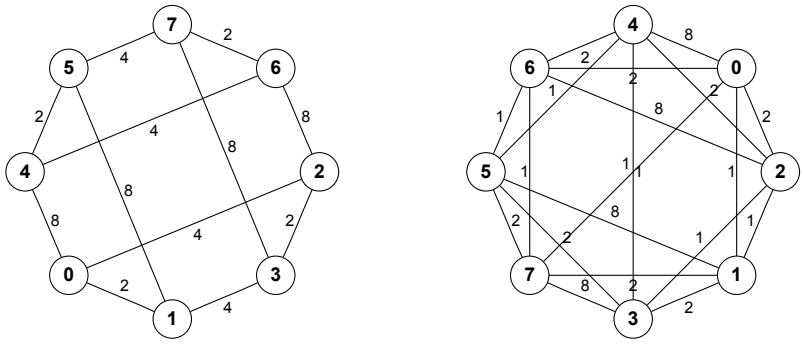
Алгоритм Дж. Брука (J. Bruck) [3] подобен алгоритму рекурсивного сдваивания, но допускает реализацию ТЦО для произвольного количества n параллельных ветвей. На шаге $k = 0, 1, \ldots, \lceil \log_2 n \rceil - 1$ ветвь i передает все принятые сообщения ветви $(i - 2^k + n) \bmod n$ и принимает сообщения 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.

Параллельная программа, реализующая коллективный обмен, может быть представлена информационным графом $G = (V, E)$, вершинам которого соответствуют ветви программы, а ребрам – информационные обмены между ними. На рис. 1 приведены графы алгоритмов рекурсивного сдваивания и Дж. Брука. Вес d_{ij} ребра отражает объем данных переданных по нему при реализации алгоритма.

3. Метод оптимизации алгоритмов коллективных обменов

Современные распределенные ВС имеют иерархическую структуру, которая может быть представлена в виде дерева, содержащего L уровней [5]. 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: N – количество ЭМ в системе; b_l – максимальное значение пропускной способности каналов связи на уровне уровня l; $z(p, q)$ – номер



(а) (б)

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

уровня функционального модуля, являющегося ближайшим общим предком для ЭМ $p, q \in \{1, 2, \ldots, N\}$, иначе говоря, номер уровня, через который они взаимодействуют.

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; $q_1, q_2, \ldots, q_h$ – номера выделенных функциональных модулей; s_r – количество ЭМ функционального модуля q_r, выделенных для реализации программы, $r \in \{1, 2, \ldots, h\}$.

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. Рассмотрим основные шаги метода, осуществляемые до реализации ТЦО.

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 $R = (r_0, r_1, \ldots, r_{n-1})$ информационного графа $G = (V, E)$ на h непересекающихся подмножеств $V_1, V_2, \ldots, V_h$ с целью минимизации суммарного веса рёбер, инцидентных различным подмножествам 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 $E' = \{(i, j) \in E | r_i \neq r_j, i \neq j\}$. Тогда вес ребра $(i, j) \in E$, инцидентного вершинам из разных подмножеств $i \in V_u$ и $j \in V_v$, есть $d_{ij}/b_{z(x_i,x_j)}$. Разбиение R должно доставлять минимум целевой функции $F(r_0, r_1, ..., r_{n-1}) = \sum_{(i,j)\in E'} d_{ij}/b_{z(x_i,x_j)}$.

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. 2 показано применение описанного метода для алгоритма Дж. Брука на кластерной ВС с иерархической структурой. Разработаны версии алгоритмов рекурсивного

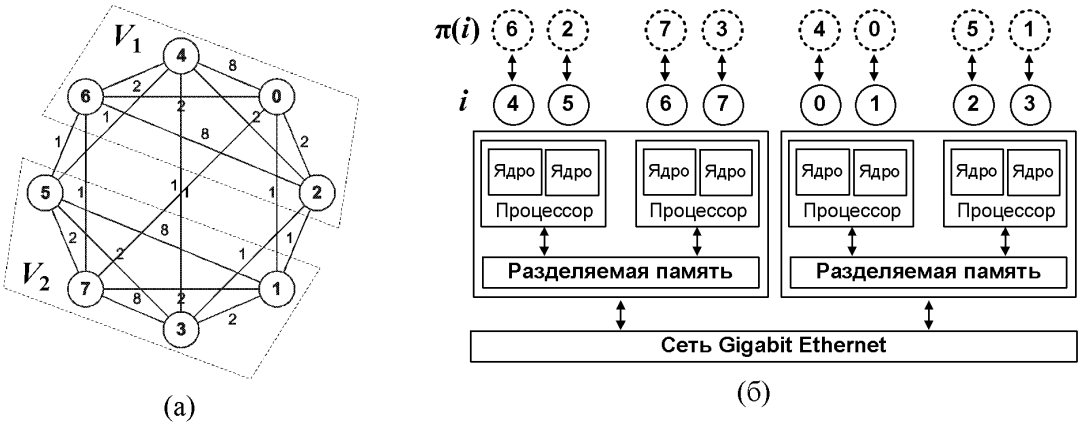


Рис. 2: Пример оптимизации алгоритма Дж. Брука ($L = 2; n = 8; h = 2; s_1 = 4; s_2 = 4$): *а* - разбиение графа алгоритма Дж. Брука на подмножества V_1 и V_2; *б* - отображение $\pi(i)$ номеров ветвей программы

сдваивания и Дж. Брука, обеспечивающие реализацию ТЦО с заданными распределениями $\pi(i)$ и $\pi^{-1}(i)$ 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 $k = 0, 1, ..., \lceil \log_2 n \rceil - 1$ 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 $k = 0, 1, ..., \lceil \log_2 n \rceil - 1$ 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 $j = 0, 1, ..., n - 1$ 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 $k = 0, 1, ..., \log_2 n - 1$ 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 $k = 0, 1, ..., \log_2 n - 1$ ветвь i и $\pi^{-1}(i \oplus 2^k)$ обмениваются ранее принятыми 2^k сообщениями. Далее каждая ветвь переставляет сообщение из позиции $j = 0, 1, ..., n - 1$ 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.

4. Результаты моделирования

Исследование созданных алгоритмов проводилось на вычислительных кластерах, укомплектованных многопроцессорными узлами и коммуникационными сетями на базе тех-

нологий Gigabit Ethernet и InfiniBand 4x DDR.

На рис. 3 представлено среднее время ТЦО "короткими" сообщениями на подсистеме из 64 процессорных ядер (4 вычислительных узла, сеть Gigabit Ethernet). Время построения отображений $\pi(i)$ и $\pi^{-1}(i)$ для примера на рис. 3 составило 480 мкс и 300 мкс, соответственно, для алгоритмов Bruck exch., Bruck reorder и recursive doubling exch., recursive doubling reorder. Накладные расходы на построение отображений не значительны и компенсируются сокращением времени реализации ТЦО созданными алгоритмами.

На рис. 4 приведены значения коэффициентов ускорения созданных алгоритмов при передаче 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.

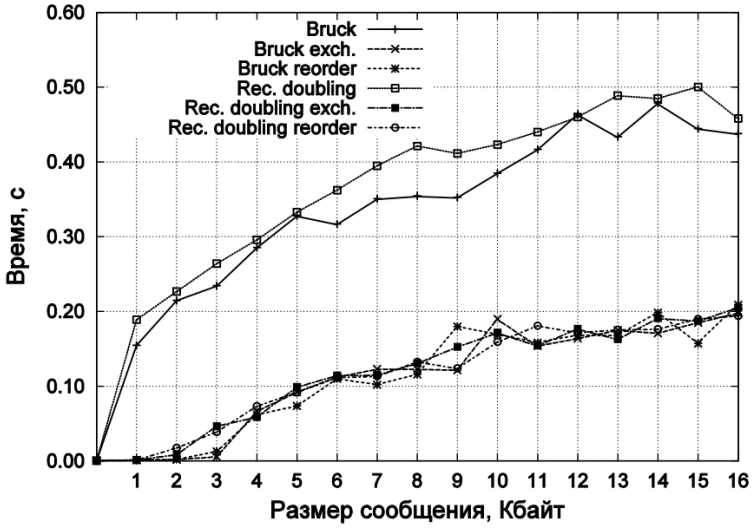


Рис. 3: Зависимость времени ТЦО от размера m сообщения на кластерной ВС ($n = 64; h = 8; s_1 = s_2 = \ldots = s_8 = 8$)

Ускорение алгоритмов реализации ТЦО между 64 параллельными ветвями на вычислительном кластере с коммуникационной сетью на базе InfiniBand 4x DDR наблюдается только для сообщений размеров 256 байт – 128 Кбайт. Это объясняется тем, что при передаче сообщений размеров больше 4 Мбайт сеть InfiniBand превосходит общую память узла по времени передачи данных.

Разработанные 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.

5. Заключение

Предложенный метод позволяет оптимизировать по времени выполнения алгоритмы коллективных обменов информацией между ветвями параллельных программ при их

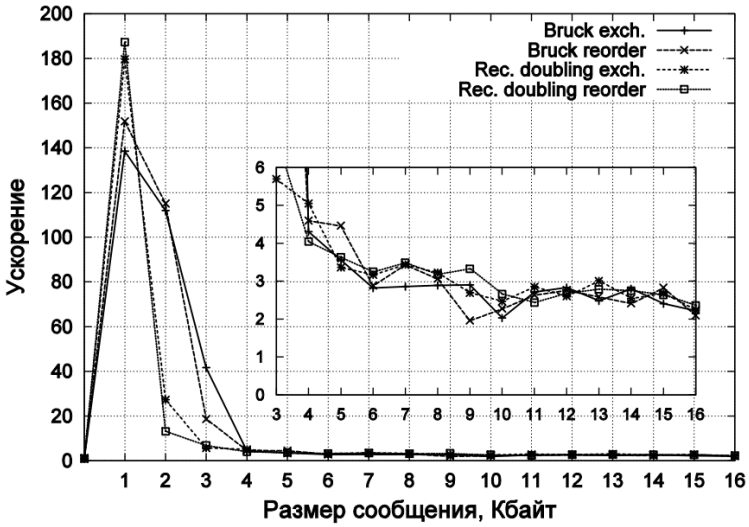


Рис. 4: Зависимость коэффициента ускорения алгоритмов ТЦО от размера m сообщения на кластерной ВС ($n = 64; h = 8; s_1 = s_2 = \ldots = s_8 = 8$)

реализации на распределенных вычислительных системах с иерархической структурой. Моделирование оптимизированных версий алгоритмов рекурсивного сдваивания (Recursive Doubling) и Дж. Брука (J. Bruck) на вычислительных кластерах с многоядерными процессорами и коммуникационными средами на базе технологий Gigabit Ethernet и InfiniBand 4x DDR подтвердили превосходство во времени реализации ТЦО созданными алгоритмами по сравнению с их исходными версиями. Накладные расходы на оптимизацию незначительны и компенсируются сокращением времени при многократном обращении к функциям реализации ТЦО. Разработанные 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.

Список литературы

[1] Хорошевский В.Г. Архитектура вычислительных систем. Москва: МГТУ им. Н.Э. Баумана, 2008. 520 с.

[2] Rabenseifner R. Automatic MPI Counter Profiling // 42nd Cray User Group. Noorwijk, The Netherlands, 2000.

[3] Thakur R., Rabenseifner R., and Gropp W. Optimization of Collective Communication Operations in MPICH // International Journal of High Performance Computer Applications. 2005. Vol. 19, No. 1. P. 49—66.

[4] Balaji P., Buntinas D., Goodell D., Gropp W., Kumar S., Lusk E., Thakur R., and Traff J.L. MPI on a Million Processors // 16th PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface. Berlin, 2009. LNCS, Vol. 5759. P. 20–30.

[5] Khoroshevsky V., Kurnosov M. Mapping Parallel Programs into Hierarchical Distributed Computer Systems // Proceedings of 4th International Conference Software and Data Technologies. INSTICC, Portugal, 2009. Vol. 2. P. 123–128.