УДК 004.021

Анализ и оптимизация конвейерных алгоритмов широковещательной передачи стандарта MPI

М. Г. Курносов*

Выполнен теоретический и экспериментальный анализ времени выполнения древовидных алгоритмов операции `MPI_Bcast`. В модели Хокни определены оптимальные степени деревьев и размеры сегментов в конвейеризированных версиях алгоритмов. Исследование алгоритмов выполнено с учетом деталей их реализации в библиотеки Open MPI. Проведено экспериментальное исследование масштабируемости алгоритмов на вычислительном кластере с сетью связи Gigabit Ethernet.

Ключевые слова: широковещательная передача, трансляционный обмен, broadcast, MPI, параллельное программирование, вычислительные системы.

1. Введение

Со второй половины 1990-х годов основным средством создания параллельных программ, переносимых между широким спектром архитектур вычислительных систем (ВС), являются библиотеки стандарта MPI (message passing interface) [1]. Базовым средством взаимодействия параллельно выполняющихся MPI-процессов является передача сообщений по каналам связи. Стандартом определены три типа информационных обменов между процессами MPI-программы: *двусторонние обмены* (point-to-point communications), односторонние обмены (one-sided communications, remote memory access) и *коллективные обмены* (групповые, глобальные, collective communications). В двустороннем обмене участвуют два процесса путем явного обращения к функциям передачи (`MPI_Send`) и приема сообщения (`MPI_Recv`). Побочным эффектом этого вида информационного взаимодействия является синхронизация выполнения пары процессов, что может оказать негативное влияние на масштабируемость параллельной программы. При одностороннем обмене в операцию вовлечен только один процесс, который выполняет удаленную операцию записи (`MPI_Put`) или чтения (`MPI_Get`) области памяти другого процесса. Синхронизации процессов не происходит, пока соответствующую операцию явно не вызовет один из процессов. Коллективные операции являются самыми ресурсоемкими, в них участвуют все процессы коммуникатора параллельной программы. Стандарт MPI 3.1 определяет 17 коллективных операций, которые можно разделить на три группы: трансляционный обмен (ТО, one-to-all broadcast/scatter), коллекторный (КО, all-to-one gather) и трансляционно-циклический (ТЦО, all-to-all/allgather).

В данной работе основное внимание сосредоточено на алгоритмах реализации операции трансляционной передачи сообщения (one-to-all broadcast) из заданного корневого процесса всем остальным. Данная операция реализуется функцией

```
MPI_Bcast(buf, count, datatype, root, comm),
```

которая выполняет передачу из памяти процесса `root` буфера `buf`, содержащего `count` элементов типа `datatype`, всем процессам коммуникатора `comm`.

Производительность (время выполнения) операции `MPI_Bcast` является критически важной для широкого спектра параллельных алгоритмов и программ. В табл. 1 приведены паке-

*Работа выполнена при поддержке РФФИ (проект № 18-07-00624) и частичной финансовой поддержке Президиума РАН (ГЗ 0306-2018-0012).

ты суперкомпьютерного моделирования, производительность которых в значительной степени зависит от времени выполнения операции `MPI_Bcast`. Результаты получены путем анализа отчетов международного проекта HPC Advisory Council [2, 3]. Функции перечислены в порядке убывания частоты вызовов и суммарного времени их выполнения. Для ряда пакетов приведены значения размеров сообщений, характерные для тестовых наборов данных, на которых выполнялось профилирование пакетов.

Таблица 1. Пакеты параллельного моделирования, производительность которых значительно зависит от времени выполнения операции `MPI_Bcast`

Пакет	Предметная область	Размеры сообщений
Abaqus	Инженерный анализ методом конечных элементов	64-256 байт
ANSYS CFX, FLUENT	Вычислительная аэро-, гидро- и газовая динамика	< 64 Кбайт
FLOW-3D	Вычислительная аэро-, гидро- и газовая динамика	4-16 байт, 1-4 Мбайт
HPCC FFT	Быстрое преобразование Фурье (1D)	
HPCG	Решение систем линейных уравнений с разреженной матрицей	< 2 Кбайт
LAMMPS	Молекулярная динамика	< 256 Кбайт
LS-DYNA	Анализ процессов в задачах механики твердого и жидкого тела	< 4 Кбайт
VASP	Квантово-механическое моделирование	< 256 Кбайт
WRF	Моделирования климатических и погодных явлений	< 16 Кбайт

Существует две основные свободно распространяемые реализации стандарта MPI, Open MPI и MPICH, на базе которых созданы все известные библиотеки MPI, включая коммерческие пакеты ведущих производителей ВС:

- Open MPI: Mellanox HPCX, IBM Spectrum MPI, Fujitsu Post-K MPI, Bull MPI;
- MPICH: MVAPICH, Intel MPI, Tianhe2 TH-MPI, Cray MPI, IBM PE MPI.

Для каждой коллективной операции в библиотеке MPI присутствует несколько алгоритмов ее реализации, обладающих разной эффективностью: архитектурно-ориентированные алгоритмы (topology-aware collectives) и универсальные алгоритмы на основе операций `MPI_Send`/`MPI_Recv`. Архитектурно-ориентированные алгоритмы имеют высокую эффективность за счет узкой специализации для фиксированной коммуникационной технологии (Mellanox InfiniBand CORE-Direct) или коммуникационной сети ВС (IBM BlueGene/Q, Cray Aries CE, Fujitsu Tofu). Универсальные алгоритмы коллективных операций основаны на использовании только операций `MPI_Send` и `MPI_Recv`, которые реализуются нижележащим слоем библиотеки MPI. Такие алгоритмы учитывают только общее число процессов в программе и параметры коллективной операции (размер сообщения, тип данных, номер корневого процесса и др.). Последнее обеспечивает их широкую переносимость между различными архитектурами ВС и сохранение приемлемого для практики уровня производительности.

При вызове коллективной операции библиотека MPI динамически определяет для заданного MPI-коммуникатора (числа процессов) и параметров операции (размера сообщения, типа данных, номера корневого процесса), какой из доступных алгоритмов широковещательной передачи использовать. В текущих реализациях MPICH, MVAPICH, Open MPI используются либо фиксированные правила выбора алгоритма на основе размера сообщения, либо выбранный пользователем алгоритм. Значительная часть алгоритмов операции `MPI_Bcast` имеет настраи-

ваемые параметры, влияющие на общее время выполнения операции.

Цель данной работы – выполнить теоретический и экспериментальный анализ времени выполнения древовидных алгоритмов операции `MPI_Bcast` и определить оптимальные значения настраиваемых параметров алгоритмов – степени деревьев и размеры сегментов в конвейеризированных версиях алгоритмов. В отличие от ранее выполненных работ [4, 5, 8, 9], в данной работе анализ времени выполнения алгоритмов не ограничивается построением асимптотических оценок, уделено внимание и константам при параметрах α и β используемой модели Хокни. Теоретические оценки подкреплены результатами экспериментов на вычислительной системе с сетью связи стандарта Gigabit Ethernet. Алгоритмы операции `MPI_Bcast` исследованы с учетом деталей их реализации в библиотеки Open MPI 4.0.0, в разработке которой автор принимал участие.

Организация работы: в следующем разделе дается описание модели Хокни, в рамках которой осуществляется анализ временных характеритик алгоритмов коллективных операций. Далее приводится описание и анализ основных алгоритмов – от простейших к наиболее эффективным. В последних разделах даются рекомендации по использованию алгоритмов.

2. Модель Хокни

Модель Хокни (R. Hockney) – одна из простейших линейных моделей, описывающая вычислительную систему двумя параметрами α и β. Время $t(m)$ передачи сообщения размером m байт между двумя процессами выражается как

$$t(m) = \alpha + m\beta,$$

где α – задержка при передаче сообщения (латентность, latency, startup time), β – время, требуемое для передачи одного байта сообщения. Также данная модель известна под названием «latency-bandwidth model». Параметр α включает в себя как латентность канала связи, так и накладные расходы процессора на передачу/прием сообщения (например, упаковка сообщения в структуры данных протокола MPI, инициализация канала связи). Величина $1/\beta$ отражает пропускную способность канала связи (bandwidth). Как правило, за ее значение принимается верхняя граница пропускной способности для конкретной коммуникационной технологии (например, Gigabit Ethernet, InfiniBand FDR).

На рис. 1 показана иллюстрация модели Хокни. Процесс 0 три раза передает сообщени размером m байт процессу 1. Процесс 1 получает первый байт первого сообщения через α единиц времени и принимает все сообщения за время $m\beta$. Процесс 0 не может начать передачу очередного сообщения до тех пор, пока процесс 1 не получил предыдущее. Поэтому второе сообщение процесс 0 начинает передавать в момент времени $\alpha + m\beta$, а процесс 1 получает его в момент времени $2(\alpha + m\beta)$. Оба процесса заканчивают обмены за время $3(\alpha + m\beta)$.

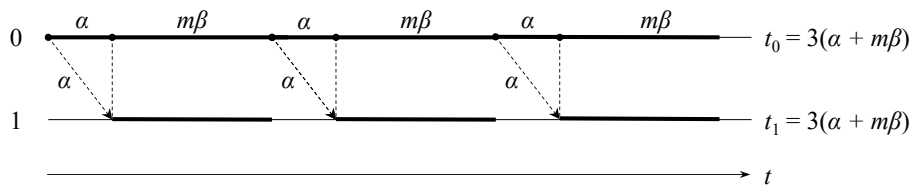


Рис. 1. Модель Хокни: процесс 0 последовательно передает три сообщения размером m байт процессу 1

Модель максимально абстрагируется от архитектуры ВС и деталей реализации инструментария параллельного программирования (игнорируется многопоточность, присутствие неблокирующих коммуникационных операций). Предполагается, что латентность α и пропускная

способность $1/\beta$ не зависят от размера m передаваемого сообщения. Считается, что каналы связи допускают двусторонний обмен данными (bidirectional channels). Коммуникационная сеть является надежной и имеет полносвязную логическую топологию (fully connected). Каждый процесс может выполнять надежные двусторонние обмены операциями `MPI_Send` и `MPI_Recv` с любым другим процессом. Системное устройство элементарной машины является однопортовым (single-ported) – каждая элементарная машина ВС (процессор) подключена к сети через один коммуникационный канал. Время передачи информации между любой парой элементарных машин одинаково.

3. Линейный алгоритм

В *линейном алгоритме* (linear broadcast, flat tree) корневой процесс с номером *root* последовательно передает сообщение процессам $0, 1, \ldots, p-1$. На рис. 2 показан псевдокод алгоритма, который выполняют все процессы программы. Алгоритм имеет низкую масштабируемость – время его выполнения линейно зависит от числа p процессов. Из временной диаграммы рис. 3 видно, что наибольшим временем выполнения характеризуется корневой процесс (при любом выборе значения $root$), отсюда время работы линейного алгоритма

$$t = (p-1)(\alpha + m\beta) = O(p).$$

```
procedure BcastLinear(buf, count, root)
  if rank = root then
    for i = 0 to p-  1 do
      if i != root then
        Send(buf, count, i)
      end if
    end for
  else
    Recv(buf, count, root)
  end if
end procedure
```

Рис. 2. Псевдокод линейного алгоритма Bcast

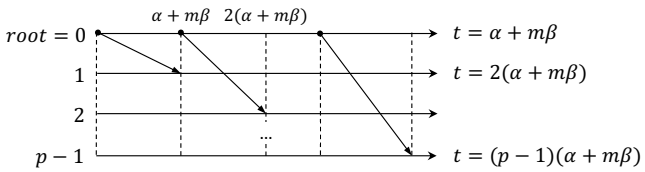


Рис. 3. Временная диаграмма выполнения линейного алгоритма

В линейном алгоритме наименьшим временем t_{min} выполнения характеризуется процесс 0 или 1 (в зависимости от выбора корневого процесса):

$$t_{min} = \alpha + m\beta.$$

Разность между временем выполнения самого медленного процесса $(p-1)(\alpha + m\beta)$ и самого быстрого $\alpha + m\beta$ линейно зависит от числа процессов, что свидетельствует о значительной разбалансированности времени работы процессов. Последнее негативно сказывается на общем балансе времени выполнения ветвей параллельной программы. Такие алгоритмы не рекомендуется использовать в итерационных вычислительных методах, их использование ограничено случаями, когда другие алгоритмы недоступны по разным причинам (например, не допускают реализацию для заданного размера сообщения или числа процессов). Значительно лучшими динамическими характеристиками обладает класс алгоритмов широковещательной передачи, основанных на логическом выстраивании процессов в деревья различных видов и параллельной передаче сообщений в поддеревьях. Выполним анализ этих алгоритмов и определим их оптимальные параметры в модели Хокни.

4. Алгоритм k-арного дерева

В рассматриваемом алгоритме процессы программы логически выстраиваются в *k-арное дерево* (k-ary tree). Каждый процесс, зная свой номер $rank$ и общее число p процессов, определяет номер своего родителя и номера не более k дочерних процессов. Основная идея алгоритма – реализовать параллельную передачу сообщения (его ретрансляцию) через промежуточные узлы поддеревьев. Все процессы, за исключением корневого, на первом шаге выполняют прием сообщения от родительского процесса, после этого выполняют последовательную передачу сообщения дочерним процессам, начиная с левого. Например, в бинарном дереве (рис. 4.*а*) корневой процесс на первом шаге передает сообщение левому дочернему процессу, а на втором шаге – правому (номер шага отмечен над ребром). Одновременно на втором шаге левый дочерний процесс корня передает принятое сообщение своему левому дочернему узлу. На рис. 4.*б* и рис. 4.*в* приведены примеры тернарного и k-арного деревьев, а также показаны параметры деревьев и оценки сверху времени t выполнения алгоритмов.

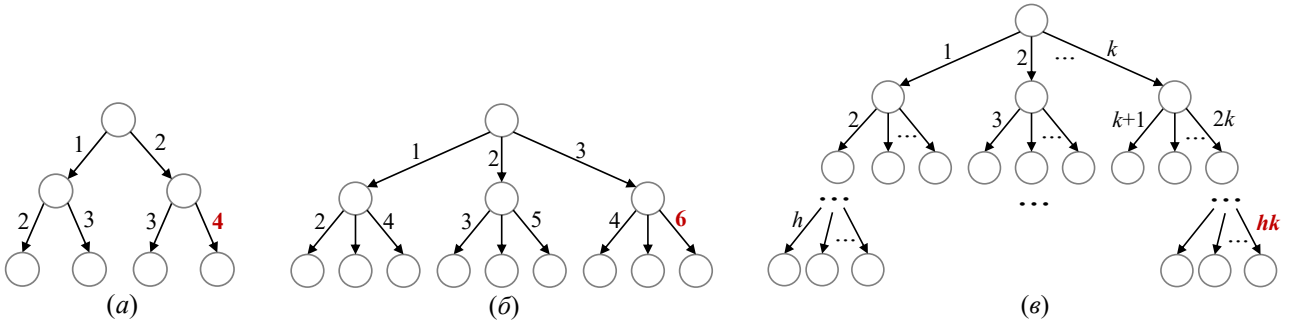


Рис. 4. Структура совершенных k-арных деревьев широковещательной передачи:
а – бинарное дерево (binary tree, $k = 2, p = 7, h = 2$): $t = 2\lfloor \log_2 p \rfloor (\alpha + m\beta)$;
б – тернарное дерево (ternary tree, $k = 3, p = 13, h = 2$): $t = 3\lfloor \log_3 p \rfloor (\alpha + m\beta)$;
в – k-арное дерево (k-ary tree): $t = k\lfloor \log_k p \rfloor (\alpha + m\beta)$

Заметим, что поиск процессом своего положения в дереве в худшем случае требует порядка $O(\log p)$ арифметико-логических операций. Как правило, подобные вычисления параметров деревьев выполняются только на этапе создания MPI-коммуникаторов и сохраняются в памяти каждого процесса.

Оценим время выполнения алгоритма в модели Хокни. Для заданного числа p процессов k-арное дерево может быть построено различными способами: иметь вид завершенного k-арного дерева (complete k-ary tree) либо k-арного дерева, сбалансированного по числу процессов в поддеревьях. Последняя форма дерева наряду с другими используется в библиотеке Open MPI (`coll/base`) для реализации операций `MPI_Bcast` и `MPI_Reduce`. Указанные деревья отличаются схемой заполнения листьями последнего уровня. В завершенном дереве узлы добавляются слева направо, все внутренние вершины имеют степень k (рис. 5.*а*). В сбалансированном по числу узлов дереве вершины добавляются слева направо, но к родителям из разных поддеревьев. Например, на рис. 5.*б* листовые узлы равномерно распределены по поддеревьям с корнями в узлах 1, 2 и 3.

Если дерево из p процессов имеет вид совершенного k-арного дерева (perfect k-ary tree), в котором все внутренние узлы имеют степень k, а листья находятся на одном уровне h, то время t_{max} выполнения алгоритма определяется временем работы самого правого листового процесса (рис. 4.*в*):

$$t_{max} = kh(\alpha + m\beta) = k\lfloor (\log_k(p(k-1)+1) - 1) \rfloor (\alpha + m\beta).$$

Рис. 5. Тернарное дерево широковещательной передачи ($p = 11, root = 0$):
а – завершенное тернарное дерево;
б – тернарное дерево, сбалансированное по числу узлов в поддеревьях

Наименьшим временем выполнения характеризуется самый левый лист

$$t_{min} = h(\alpha + m\beta) = \lceil (\log_k(p(k-1)+1) - 1) \rceil (\alpha + m\beta).$$

Если процессов не хватает для формирования совершенного вида дерева, то время t_{max} выполнения может варьироваться в следующем интервале:

$$k(h-1)(\alpha + m\beta) \le t_{max} \le kh(\alpha + m\beta).$$

Например, на рис. 4 для формирования совершенного дерева не хватает двух процессов, здесь наибольшее время выполнения имеет процесс 9. Далее, не теряя общности, будем полагать, что процессы образуют совершенное дерево. Указанное допущение не влияет на дальнейшие рассуждения, так как при анализе времени выполнения алгоритма нас интересуют оценки сверху времени выполнения. Тогда

$$t = k\lfloor \log_k p \rfloor (\alpha + m\beta) = O(\log p).$$

На практике при реализации алгоритмов и разработке библиотек MPI встает вопрос о выборе оптимального значения степени k дерева для заданного числа p процессов. Такое решение принимается динамически при формировании MPI-коммуникаторов. В библиотеках Open MPI и MVAPICH по умолчанию используется значение $k = 4$ для любого числа процессов. Определим оптимальное в модели Хокни значение k. Запишем оценку сверху времени выполнения алгоритма как функцию от параметра k и найдем его значение, при котором время выполнения алгоритма минимально:

$$t(k) = k \log_k p(\alpha + m\beta), \quad \frac{\partial t(k)}{\partial k} = \frac{\ln k - 1}{\ln^2 k} = 0, \quad k = e \approx 2.7183.$$

Таким образом, наименьшим в модели Хокни среди k-арных деревьев временем выполнения обладает тернарное дерево (k = 3). Полученная оценка хорошо согласуется с практикой – минимальное время выполнения достигается на k-арных деревьях небольших степеней. С увеличением значения k наблюдается рост времени выполнения алгоритма. На рис. 5 показаны результаты измерения времени выполнения алгоритма k-арного дерева широковещательной рассылки для различных значений k. Измерения выполнены для реализации алгоритма в библиотеке Open MPI 4.0.0 общепринятым тестом Intel MPI Benchmarks (приведено среднее время выполнения по не менее 40 запускам для каждого k) для 128 процессов: 16 двухпроцессорных узлов (2 x Intel Xeon Quad), сеть связи – Gigabit Ethernet.

В отличие от линейного, рассматриваемый алгоритм хорошо масштабируется – время его работы логарифмически зависит от числа процессов. В то же время алгоритм имеет существенный недостаток – фиксированная степень k корня и внутренних узлов является причиной их недостаточной загруженности операциями передачи. Так, в тернарном дереве (рис. 4.*б*)

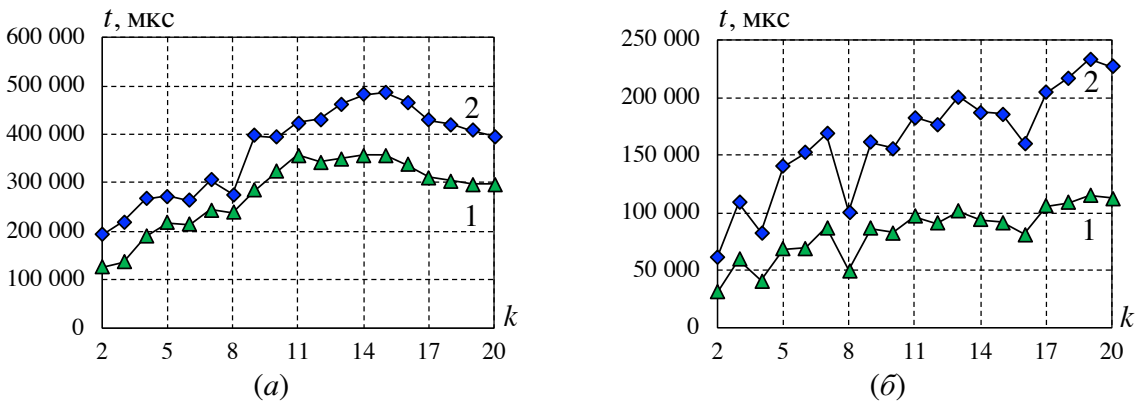


Рис. 6. Зависимость времени выполнения алгоритма от степени k дерева ($p = 128$, 1 – размер сообщения 1 МБ; 2 – размер сообщения 2 МБ):
а – алгоритм k-арного дерева;
б – алгоритм k-номиального дерева

после выполнения трех операций корень завершает свою работу, хотя в это время в дереве присутствуют узлы, ожидающие сообщения. Указанного недостатка лишен алгоритм k-номиального дерева.

5. Алгоритм k-номиального дерева

В алгоритме k-номиального дерева процессы логически выстраиваются в дерево обменов, представляющее совокупность k-номиальных деревьев (k-nomial tree). При $k = 2$ дерево называется *биномиальным* (binomial tree), а при $k = 3$ – *триномиальным* (trinomial tree). На рис. 7 показаны примеры биномиального, 3-номиального и 4-номиального деревьев для широковещательной передачи сообщения из корневого процесса 0. Каждый процесс, за исключением корневого, выполняет два шага – прием сообщения от родителя и его передачу дочерним процессам, начиная с самого левого. Номера процессов трактуются как числа, записанные в системе счисления с основанием k. Номер родительского процесса вычисляется путем сброса в номере текущего процесса самого младшего разряда с положительным весом (least significant digit, LSD). Например, в биномиальном дереве номер родителя для процесса $12 = 1100_2$ равен $8 = 1000_2$, который получен путем сброса в числе 12 третьего бита (младшего значащего разряда). Количество дочерних узлов любого процесса не превышает $(k-1)\lceil \log_k p \rceil$. Таким образом, в отличие от k-арных деревьев, в k-номиальных степень корня (внутреннего узла) не фиксирована, а логарифмически зависит от числа вершин в его поддереве. Последнее позволяет k-номиальным деревьям эффективнее загрузить корень и внутренние процессы передачей сообщений. Из примера на рис. 7 видно, что наименьшим временем выполнения характеризуется биномиальное дерево, которому требуется 4 шага для завершения операции.

В общем случае время $t(k)$ выполнения алгоритма определяется временем работы корневого процесса – временем завершения передачи сообщения самому правому дочернему узлу

$$t(k) = (k-1)\lceil \log_k p \rceil (\alpha + m\beta).$$

Монотонное возрастание функции $t(k)$ позволяет сделать вывод об оптимальности биномиальных деревьев в модели Хокни. Результаты экспериментов на вычислительных кластерах подтверждают оптимальный характер биномиальных деревьев. На рис. 6.*б* показано монотонное возрастание времени выполнения алгоритма k-номиального дерева с увеличением степени дерева на вычислительном кластере с сетью связи Gigabit Ethernet.

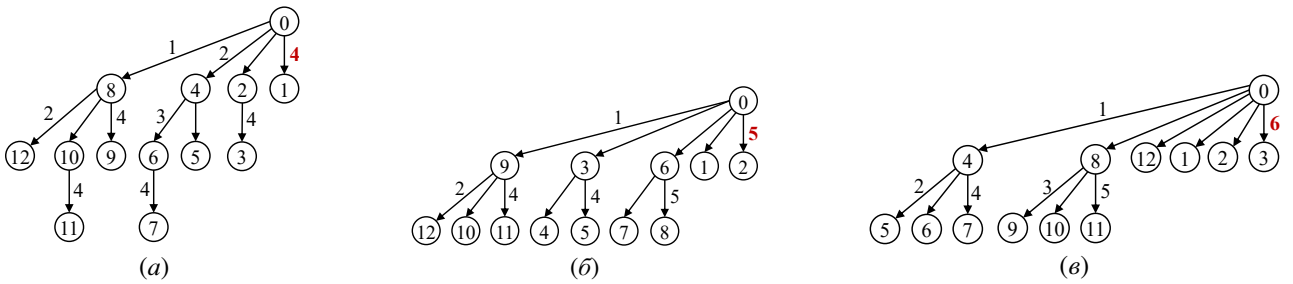


Рис. 7. Структура k-номиальных деревьев широковещательной передачи:
а – биномиальное дерево (binomial tree, $k = 2, p = 13, h = 3$);
б – триномиальное дерево (trinomial tree, $k = 3, p = 13, h = 2$);
в – 4-номиальное дерево (4-nomial tree, $k = 4, p = 13, h = 2$)

Несмотря на указанные достоинства, данный алгоритм не учитывает возможность одновременного приема и передачи сообщений одним процессом по дуплексным каналам связи, которые характерны для подавляющего большинства коммуникационных сетей ВС (Gigabit Ethernet, InfiniBand). Возможным решением является разработка алгоритма, в котором исходное сообщение разбивается на сегменты и передается по конвейеру, так, чтобы промежуточные процессы одновременно передавали текущий сегмент и принимали следующий. Рассмотрим этот подход на примере анализа и оптимизации конвейерного алгоритма для больших сообщений.

6. Конвейерный алгоритм

В *конвейерном алгоритме* (linear pipeline) исходное сообщение длиной m байт разбивается на $\lceil m/s \rceil$ сегментов по s байт, которые по цепочке передаются от процесса к процессу. Промежуточные процессы после получения очередного сегмента m_i начинают использовать дуплексный режим передачи – передавать текущий сегмент m_i и одновременно с этим принимать следующий сегмент m_{i+1}. На рис. 8 приведена диаграмма выполнения конвейерного алгоритма для четырех процессов и сообщения, разбитого на три сегмента m_1, m_2 и m_3. Промежуточные процессы 1 и 2 конвейера после получения сегмента m_1 сразу начинают его передавать следующему процессу и одновременно с этим принимать сегмент m_2.

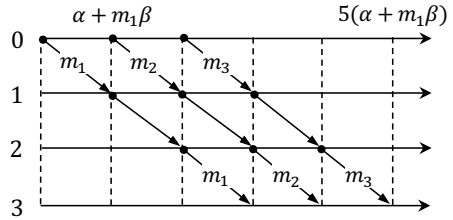


Рис. 8. Диаграмма выполнения конвейерного алгоритма ($p = 4$, три сегмента)

Наибольшим временем выполнения характеризуется процесс $p - 1$, который последним получает заключительный сегмент сообщения. Первый сегмент он получает в момент времени $(p - 1)(\alpha + s\beta)$, второй в $p(\alpha + s\beta)$, третий в $(p + 1)(\alpha + s\beta)$ и т.д. Заключительный сегмент процесс $p-1$ получит в момент времени $\left(p - 2 + \left\lceil \frac{m}{s} \right\rceil\right)(\alpha + s\beta)$. Таким образом, время работы конвейерного алгоритма

$$t(s) = \left(p - 2 + \left\lceil \frac{m}{s} \right\rceil\right)(\alpha + s\beta).$$

Определим оптимальный размер s_{opt} сегмента, при котором алгоритм характеризуется минимальным в модели Хокни временем выполнения:

$$\frac{\partial t(s)}{\partial s} = -\frac{m\alpha}{s^2} + (p-2)\beta = 0, \quad s_{opt} = \sqrt{\frac{m\alpha}{(p-2)\beta}}.$$

На рис. 9.*а* показана зависимость времени выполнения конвейерного алгоритма в модели Хони от размера сегмента (параметры $\alpha = 5 \cdot 10^{-5}$ с, $\beta = 4.7 \cdot 10^{-8}$ с получены тестом Intel MPI Benchmarks для сети связи Gigabit Ethernet). Красными точками показаны оптимальные размеры s_{opt} сегментов. Полученная теоретическая оценка s_{opt} дает хорошее приближение к размеру сегмента, эффективного для реальных коммуникационных сетей. Ее использование позволяет сократить границы поиска эффективного размера сегмента для конкретной коммуникационной технологии (Gigabit Ethernet, InfiniBand). На рис. 9.*б* показаны экспериментальные результаты – время выполнения алгоритма на вычислительном кластере с сетью связи Gigabit Ethernet и установленные границы (суб)оптимальных размеров сегментов [1536, 3072] (выделены цветом).

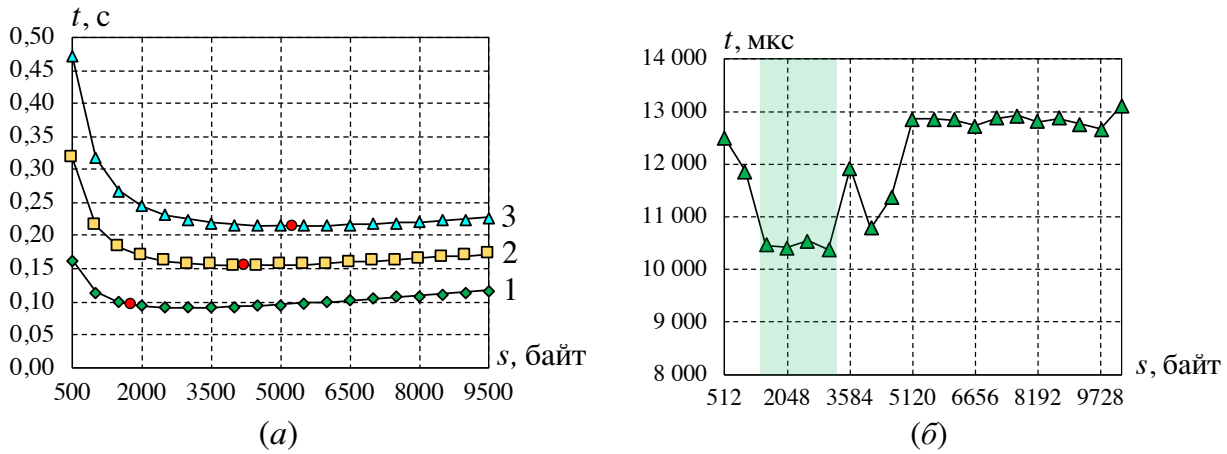


Рис. 9. Зависимость времени выполнения конвейерного алгоритма от размера s сегмента (128 процессов):
а – модели Хокни ($\alpha = 5 \cdot 10^{-5}$ с, $\beta = 4.7 \cdot 10^{-8}$ с): 1 – $m = 1$ МБ; 2 – $m = 2$ МБ; 3 – $m = 3$ МБ;
б – эксперимент на кластере (Open MPI 4.0.0, тест Intel MPI Benchmarks, сеть Gigabit Ethernet): $m = 1$ МБ

На рис. 10 показана зависимость в модели Хокни времени выполнения алгоритма биномиального дерева и конвейерного алгоритма с оптимальным размером сегмента. Для сообщений небольших размеров эффективными являются алгоритмы бинарного и биномиального деревьев. Для сообщений значительных размеров предпочтительным является конвейерный алгоритм. Очевидно, что для сообщений средних размеров разумно применить комбинированный подход – древовидный алгоритм с конвейеризацией передачи сообщений.

7. Алгоритм бинарного дерева с конвейеризацией

В алгоритме *бинарного дерева с конвейеризацией передачи сообщений* (pipelined binary tree) процессы логически организованы в бинарное дерево. Исходное сообщение длиной m байт разбивается на $\lceil m/s \rceil$ сегментов по s байт, которые передаются по дереву с учетом дуплексного режима работы каналов связи. На рис. 11 показан пример широковещательной

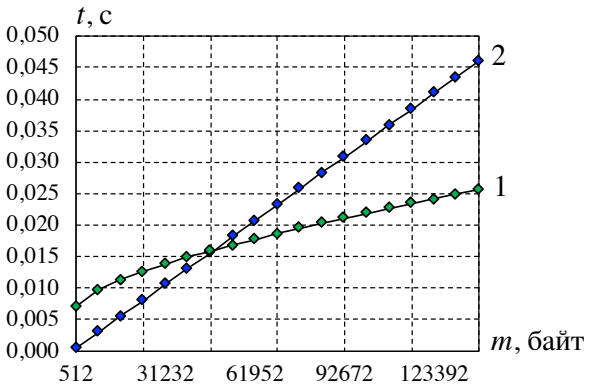
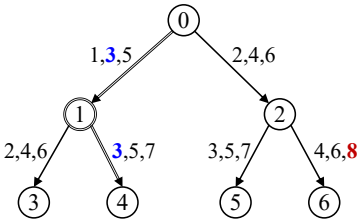


Рис. 10. Зависимость времени выполнения алгоритмов операции Bcast в модели Хокни от размера m сообщения ($\alpha = 5 \cdot 10^{-5}$ с, $\beta = 4.7 \cdot 10^{-8}$ с):
1 – конвейерный алгоритм с оптимальным размером сегмента; 2 – биномиальное дерево

передачи сообщения, разбитого на 3 сегмента. Временные шаги передачи отдельных сегментов отмечены на дугах. На шаге 3 процесс 1 начинает передачу первого сегмента процессу 4 и одновременно принимает второй сегмент от процесса 0. В таблице справа (рис. 11) подчеркнуты пары процессов, которые одновременно принимают и передают сегмент сообщения – используют дуплексный режим работы каналов связи. Как видно, не все процессы и не на каждом временном шаге в полной мере используют доступную полосу пропускания каналов связи. По этой причине алгоритму бинарного дерева с конвейеризацией передачи сообщений не удается достигнуть полной загрузки дуплексных каналов связи (full bandwidth).



Пары процессов Send→Recv	Шаг по времени 1	2	3	4	5	6	7	8
	0→1	0→2 1→3	0→1 1→4 2→5	0→2 2→6 1→3	0→1 1→4 2→5	0→2 2→6 1→3	1→4 2→5	2→6

Рис. 11. Бинарное дерево с конвейеризацией передачи сообщений ($p = 7, root = 0$, 3 сегмента, числа над дугами – временные шаги передачи сегментов)

Оценим сверху время выполнения алгоритма. Как было показано ранее для k-арных деревьев, последним первый сегмент принимает правый листовой процесс. Следующий сегмент он примет через число шагов, равное степени дерева, и т.д. Таким образом, время $t(s)$ широковещательной передачи алгоритмом бинарного дерева с конвейеризацией сообщений имеет вид

$$t(s) = 2\left(\lceil \log_2 p \rceil + \frac{m}{s} - 1\right)(\alpha + s\beta).$$

Рассуждая аналогично, можно получить оценку и для биномиального дерева с конвейеризацией сообщений

$$t(s) = \left(\lceil \log_2 p \rceil + \frac{m}{s} - 1\right)(\alpha + s\beta).$$

Определим оптимальный размер сегмента для алгоритма бинарного дерева с конвейеризацией передачи сообщений:

$$\frac{\partial t(s)}{\partial s} = -\frac{2m}{s^2}\alpha + 2(\log_2 p - 1)\beta = 0, \quad s_{opt} = \sqrt{\frac{m\alpha}{(\log_2 p - 1)\beta}}.$$

На рис. 12 показана зависимость времени выполнения алгоритма бинарного дерева с конвейеризацией передачи сообщений от размера s сегмента. Как видно из экспериментальных результатов, наименьшее время выполнения достигнуто при размере сегмента 13 312 байт. Теоретически установленный оптимальный размер сегмента $s_{opt} = 13\,632$ байт. В целом полученное значение дает хорошее приближение для выбора субоптимального значения сегмента на практике.

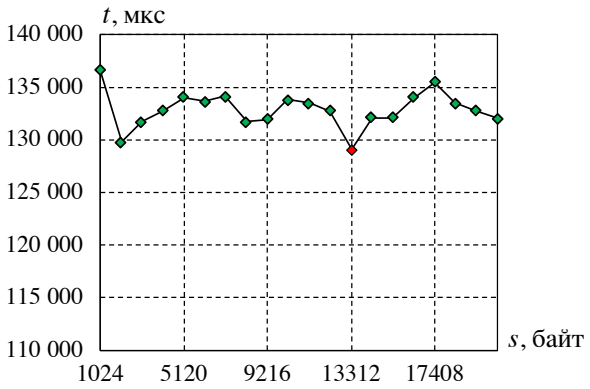


Рис. 12. Зависимость времени выполнения алгоритма бинарного дерева с конвейеризацией сообщений от размера s сегмента ($p = 128, m = 1$ МБ, Open MPI 4.0.0, тест Intel MPI Benchmarks, сеть Gigabit Ethernet)

8. Обсуждение результатов и рекомендации

Известна [9] теоретическая нижняя граница времени t_{Bcast} выполнения широковещательной передачи в модели Хокни

$$t_{Bcast} \geq \min\{\lceil \log_2 p \rceil \alpha, m\beta\}.$$

Оценим отклонение времени работы рассмотренных алгоритмов от указанной нижней границы.

Для сообщений небольших размеров ($\alpha > m\beta$) время работы алгоритма биномиального дерева определяется коэффициентом $\lceil \log_2 p \rceil$ при параметре α – вкладом латентности канала связи в суммарное время работы алгоритма

$$t_{Binomial} = \lceil \log_2 p \rceil \alpha + m \lceil \log_2 p \rceil \beta.$$

Для рассматриваемого случая сообщений небольших размеров значение коэффициента при параметре α с точностью до константы совпадает со значением нижней границы t_{Bcast}, что свидетельствует об асимптотической оптимальности алгоритма биномиального дерева для сообщений небольших размеров. При сообщениях средних и больших размеров ($\alpha < m\beta$), когда вклад латентности канала связи не оказывает решающего значения, алгоритм биномиального дерева имеет при параметре β коэффициент в $\log_2 p$ раз больший значения нижней границы. На рис. 13 приведен пример, иллюстрирующий эти два случая.

Линейная зависимость времени выполнения конвейерного алгоритма (linear pipeline) ограничивает его применение случаем незначительного числа процессов, а также сообщений больших размеров:

$$t_{Pipeline} = \left(p - 2 + \left\lceil \frac{m}{s} \right\rceil\right)(\alpha + s\beta).$$

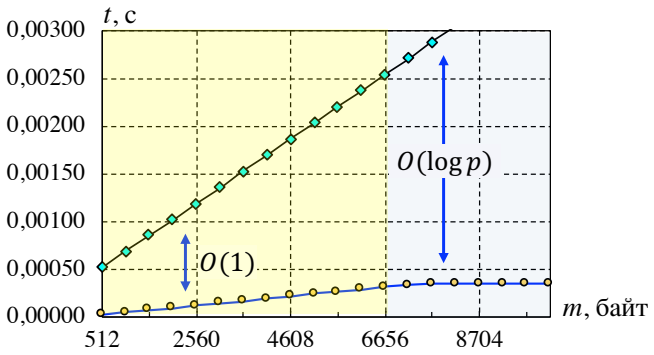


Рис. 13. Отклонение времени выполнения алгоритма биномиального дерева от нижней границы времени операции Bcast ($\alpha = 5 \cdot 10^{-5}$ с, $\beta = 4.7 \cdot 10^{-8}$ с, $p = 128, root = 0$)

Латентность алгоритма бинарного дерева с конвейеризацией сообщений (pipelined binary tree) в $2 + \frac{m/s}{\log_2 p}$ раз превышает значение латентности нижней границы

$$t_{PipelinedBinTree} = 2\left(\lceil \log_2 p \rceil + \left\lceil \frac{m}{s} \right\rceil - 1\right)(\alpha + s\beta).$$

Нетрудно заметить, что при небольших размерах сообщений и сегментов отношение времени работы алгоритма к нижней границе равно константе, что свидетельствует об асимптотической оптимальности алгоритма для данного случая. Пропускная способность алгоритма в $2 + \frac{2s \log_2 p}{m}$ раз превышает значение пропускной способности нижней границы. При больших размерах сообщений отношение равно константе, что говорит об асимптотической оптимальности алгоритма по пропускной способности.

На рис. 14 в модели Хокни показаны диапазоны размеров сообщений и рекомендуемые к использованию на них алгоритмы широковещательной передачи. Для сообщений небольших размеров рекомендуется использовать алгоритм биномиального дерева. Для сообщений средних размеров уже целесообразно задействовать конвейеризацию передачи сообщений и алгоритм бинарного или биномиального дерева. Для сообщений больших размеров и небольшого числа процессов рекомендуется использовать конвейерный алгоритм.

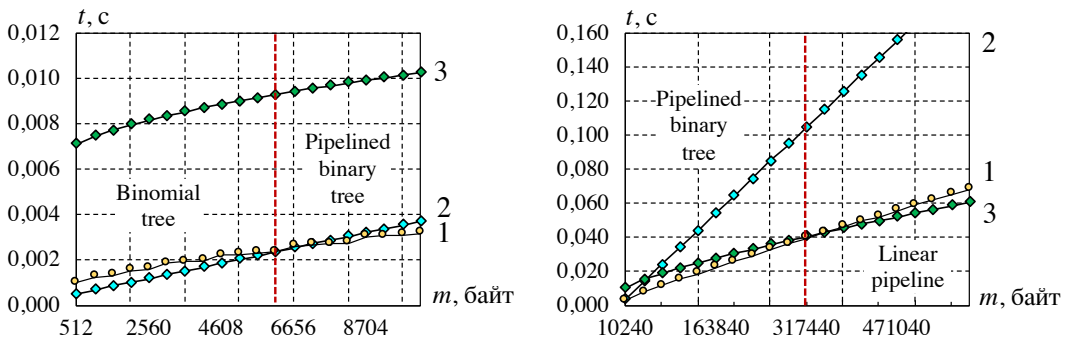


Рис. 14. Зависимость времени выполнения алгоритмов `MPI_Bcast` в модели Хокни от размера m сообщения ($\alpha = 5 \cdot 10^{-5}$ с, $\beta = 4.7 \cdot 10^{-8}$ с, $p = 128, root = 0$):
1 – алгоритм бинарного дерева с конвейеризацией (с оптимальным размером сегмента);
2 – алгоритм биномиального дерева; 3 – конвейерный алгоритм с оптимальным размером сегмента

9. Заключение

Выполненный теоретический и экспериментальный анализ времени выполнения древовидных алгоритмов операции `MPI_Bcast` показал бо́льшую эффективность алгоритма биномиального дерева по сравнению с алгоритмами тернарного и бинарного деревьев. Для алгоритмов с сегментацией сообщений установлена эффективность алгоритмов бинарного и биномиального деревьев для сообщений средних размеров. Для небольшого числа процессов и сообщений значительных размеров эффективным является конвейерный алгоритм.

Построенные аналитические выражения носят рекомендательный характер и позволяют сузить на практике поиск эффективных степеней деревьев и размеров сегментов для алгоритмов с конвейеризацией передачи сообщений.

Литература

1. MPI-3.1 Standard [Электронный ресурс]. URL: `https://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf` (дата обращения: 16.03.2019).
2. HPC Advisory Council Best Practices [Электронный ресурс]. URL: `http://hpcadvisorycouncil.com/best_practices.php` (дата обращения: 16.03.2019).
3. *Dongarra J., Beckman P., Moore T.* International Exascale Software Project Roadmap // The International Journal of High Performance Computing Applications. 2011. V. 25. P. 3–60.
4. *Thakur R., Rabenseifner R., Gropp W.* Optimization of collective communication operations in MPICH // Int. Journal of High Performance Computing Applications. 2005. V. 19 (1). P. 49–66.
5. *Bruck J. et al.* Efficient Algorithms for All-to-All Communications in Multiport Message Passing Systems // IEEE Trans. Parallel Distrib. Syst. 1997. V. 8 (11). P. 1143–1156.
6. *Курносов М. Г.* Алгоритмы трансляционно-циклических информационных обменов в иерархических распределенных вычислительных системах // Вестник компьютерных и информационных технологий. 2011. № 5. С. 27–34.
7. *Balaji P., Buntinas D., Goodell D. et al.* MPI on Millions of Cores // Parallel Processing Letters. 2011. V. 21 (1). P. 45–60.
8. *Hoefler T., Moor D.* Energy, Memory, and Runtime Tradeoffs for Implementing Collective Communication Operations // Journal of Supercomputing Frontiers and Innovations. 2014. V. 1 (2). P. 58–75.
9. *Sanders P., Speck J., Traff J. L.* Two-Tree Algorithms for Full Bandwidth Broadcast, Reduction and Scan // Parallel Computing. 2009. Vol. 35 (12). P. 581–594.

Статья поступила в редакцию 20.03.2019.

Курносов Михаил Георгиевич

д.т.н., профессор кафедры вычислительных систем СибГУТИ (630102, Новосибирск, ул. Кирова, 86), тел. (383) 269-82-86, e-mail: `mkurnosov@sibguti.ru`; старший научный сотрудник Института физики полупроводников им. А. В. Ржанова СО РАН (630090, Новосибирск, пр-т. Лаврентьева, 13), e-mail: `mkurnosov@isp.nsc.ru`.

Analysis and optimization of pipelined broadcast algorithms

M. Kurnosov

Theoretical and experemental analysis of `MPI_Bcast` algortihms are performed. The optimal tree degrees and segment sizes for pipelined versions of algorithms are obtained. Algorithms were investigated according to their implementation in the Open MPI. Theoretical results are consistent with experiments on a computer cluster with a Gigabit Ethernet communication network.

Keywords: broadcast, MPI, parallel programming, high-performance computing.