Реализация алгоритма Дж. Брука для неблокирующей операции Allgather в библиотеке LibNBC

Аненков Александр Дмитриевич

Сибирский государственный университет телекоммуникаций и информатики

E-mail: anenkov.ru@gmail.com

Ежегодная российская научно-техническая конференция

«Обработка информации и математическое моделирование»

г. Новосибирск, 2019

Коллективные обмены информацией в ВС

В коллективной операции обмена участвуют все ветви параллельной программы:

- ❑ трансляционный обмен («один-всем», one-to-all broadcast)
- ❑ коллекторный обмен («все-одному», all-to-one broadcast)
- ❑ трансляционно-циклический обмен («каждый-всем», all-to-all broadcast)

Стандарт MPI подразделяет операции коллективного обмена на:

- ❑ блокирующие
- ❑ *неблокирующие*

Коллективные операции реализованы различными алгоритмами на базе двусторонних обменов (point-to-point)

Неблокирующие операции

- Вызов функции неблокирующей коллективной операции **лишь инициализирует** её для отложного выполнения
- Непосредственный обмен сообщениями будет выполнен позднее, например, при вызове функций, проверяющих состояние операции:

```
MPI_Wait(request, status)
MPI_Waitall(count, array_of_requests, array_of_statuses)
MPI_Test(request, flag, status)
```

- Неблокирующие операции позволяют **совместить выполнение вычислений и обмен данными**

```
MPI_Request req;
MPI_Ibcast(arr, 100, MPI_INT, 0, MPI_COMM_WORLD, &req);
/* выполнение вычислений */
MPI_Wait(&req);
```

Реализации алгоритмов неблокирующих коллективных операций в библиотеках стандарта MPI

№	ОПЕРАЦИЯ	MPICH 3.3	INTEL MPI LIBRARY 2019	OPEN MPI 4.0 (LIBNBC)
1	MPI_IALLGATHER	1. Recursive doubling 2. Bruck's 3. Ring	1. Recursive doubling 2. Bruck's 3. Ring	1. Linear 2. Recursive doubling
2	MPI_IALLGATHERV	1. Recursive doubling 2. Bruck's 3. Ring (pipeline)	1. Recursive doubling 2. Bruck's 3. Ring	1. Linear
3	MPI_IALLREDUCE	1. Naive 2. Recursive doubling 3. Rabenseifner's 4. SMP	1. Recursive doubling 2. Rabenseifner's 3. Reduce + Bcast 4. Ring (patarasuk) 5. Knomial 6. Binomial 7. Reduce scatter allgather 8. SMP 9. Nreduce	1. Linear 2. Dissemination 3. Ring (segmented) 4. Rabenseifner's 5. Recursive doubling
4	MPI_IALLTOALL	1. Bruck's 2. Pairwise exchange 3. Batched sendrecv	1. Bruck's 2. Isend/Irecv + Waitall 3. Pairwise exchange	1. Linear 2. Dissemination 3. Pairwise exchange 4. Inplace
5	MPI_IALLTOALLV	1. Pairwise exchange 2. Blocked 3. Inplace	1. Isend/Irecv + waitall	1. Linear 2. Pairwise exchange 3. Inplace
6	MPI_IALLTOALLW	1. Pairwise exchange 2. Blocked 3. Inplace	1. Isend/Irecv + waitall	1. Linear 2. Pairwise exchange 3. Inplace
7	MPI_IBARRIER	1. Recursive doubling	1. Dissemination	1. Dissemination

Реализации алгоритмов неблокирующих коллективных операций в библиотеках стандарта MPI

№	ОПЕРАЦИЯ	MPICH 3.3	INTEL MPI LIBRARY 2019	OPEN MPI 4.0 (LIBNBC)
8	MPI_IBCAST	1. Binomial 2. Scatter recursive doubling allgather 3. Scatter ring allgather 4. SMP	1. Binomial 2. Recursive doubling 3. Ring 4. Knomial 5. SMP 6. Tree knominal 7. Tree kary	1. Linear 2. Chain 3. Binomial 4. Knomial
9	MPI_IEXSCAN	1. Recursive doubling	1. Recursive doubling 2. SMP	1. Linear 2. Recursive doubling
10	MPI_IGATHER	1. Linear 2. Binomial	1. Binomial 2. Knomial	1. Linear
11	MPI_IGATHERV	1. Linear	1. Linear 2. Linear ssend	1. Linear
12	MPI_IREDUCE	1. Binomial 2. Reduce scatter gather 3. SMP	1. Rabenseifner’s 2. Binomial 3. Knomial	1. Linear 2. Chain 3. Binomial 4. Rabenseifner's
13	MPI_IREDUCE_SCATTER	1. Noncommutative recursive doubling (butterfly) 2. Pairwise exchange 3. Recursive doubling 4. Recursive halving	1. Recursive halving 2. Pairwise 3. Recursive doubling	1. Pairwise exchange
14	MPI_IREDUCE_SCATTER_BLOCK	1. Noncommutative recursive doubling (butterfly) 2. Pairwise exchange 3. Recursive doubling 4. Recursive halving	--	1. Pairwise exchange
15	MPI_ISCAN	1. Recursive Doubling 2. SMP	1. Recursive Doubling 2. SMP	1. Linear 2. Recursive Doubling
16	MPI_ISCATTER	1. Binomial	1. Binomial 2. Knomial	1. Linear
17	MPI_ISCATTERV	1. Linear	1. Linear	1. Linear

Библиотека LibNBC

- Различные алгоритмы неблокирующих операций коллективного обмена реализованы в библиотеке LibNBC
- Библиотека используется в проектах Open MPI, MPICH и др.
- На сегодняшний день в библиотеке LibNBC из проекта Open MPI **отсутствуют реализации многих алгоритмов коллективных операций**, а некоторые операции представлены лишь алгоритмами с **линейной вычислительной сложностью**
- Возникают **задачи исследования** существующего набора алгоритмов неблокирующих операций коллективного обмена, **реализации новых алгоритмов** с меньшим классом сложности и **оценки их производительности**

Hoefler T., Lumsdaine A. **Design, Implementation, and Usage of LibNBC** // Open Systems Lab, Indiana University. presented in Bloomington, IN, USA, School of Informatics, Aug. 2006
Hoefler T., Lumsdaine A., Rehm W. **Implementation and Performance Analysis of Non-Blocking Collective Operations for MPI** // In Proceedings of the 2007 International Conference on High Performance Computing, Networking, Storage and Analysis, SC07, presented in Reno, USA, IEEE Computer Society/ACM, Nov. 2007

Библиотека LibNBC

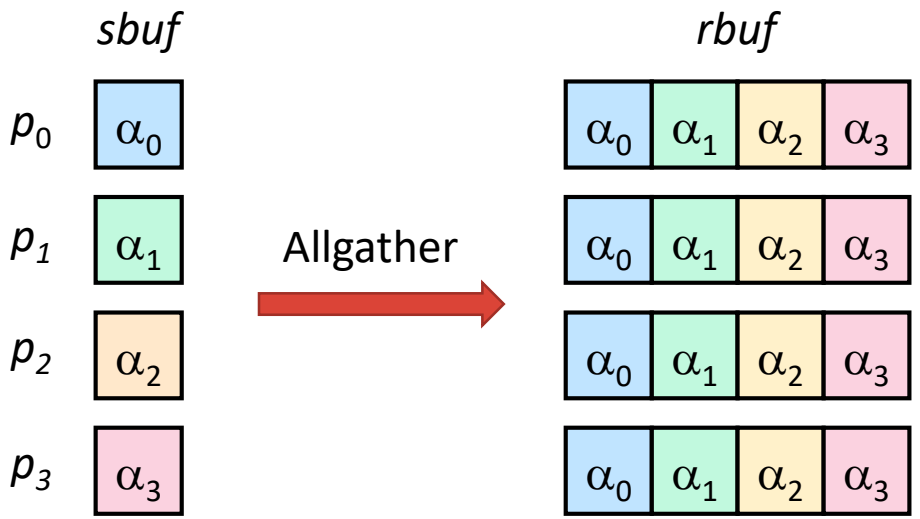
Библиотека LibNBC позволяет каждой ветви параллельной программы строить *расписание* (schedule) выполнения операций при помощи вызова следующих основных функций:

- `NBC_Sched_send` – запланировать неблокирующую операцию передачи сообщения
- `NBC_Sched_recv` – запланировать неблокирующую операцию приёма сообщения
- `NBC_Sched_copy` – запланировать операцию копирования из одного буфера в другой

Hoefler T., Lumsdaine A. **Design, Implementation, and Usage of LibNBC** // Open Systems Lab, Indiana University. presented in Bloomington, IN, USA, School of Informatics, Aug. 2006
Hoefler T., Lumsdaine A., Rehm W. **Implementation and Performance Analysis of Non-Blocking Collective Operations for MPI** // In Proceedings of the 2007 International Conference on High Performance Computing, Networking, Storage and Analysis, SC07, presented in Reno, USA, IEEE Computer Society/ACM, Nov. 2007

Коллективная операция Allgather

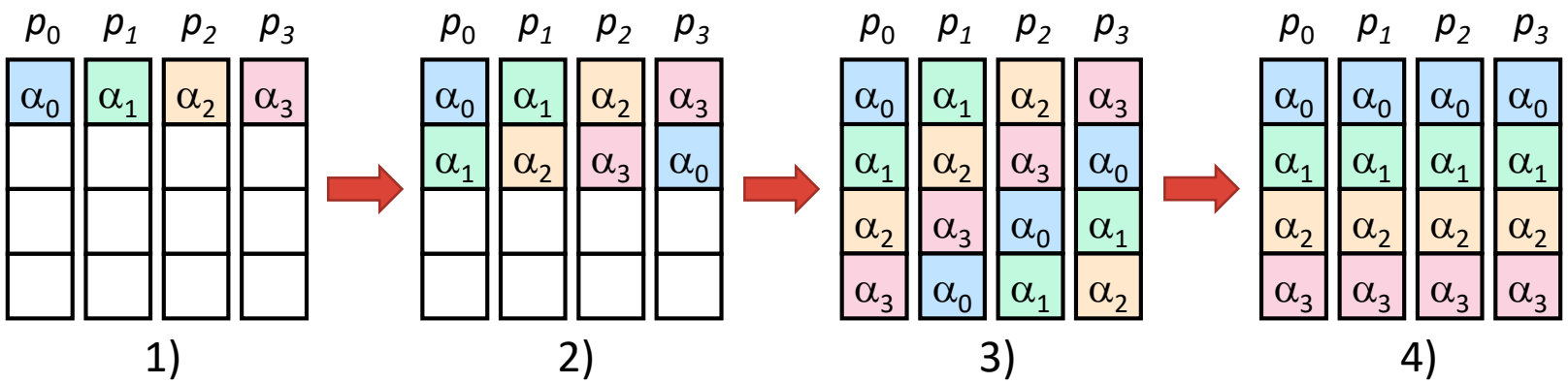
- Пусть **имеется** параллельная программа, каждая ветвь p_i которой содержит блок данных α_i размером m байт
- **Требуется** отправить блок данных α_i из буфера *sbuf* всем ветвям и получить блок данных от каждой в буфер *rbuf*
- В стандарте MPI такую задачу коллективного обмена решает **операция Allgather**



- После завершения операции содержимое буферов приёма *rbuf* во всех ветвях программы одинаково
- Длина блоков данных α предполагается одинаковой
- Операция **может быть реализована различными алгоритмами**

Алгоритм Дж. Брука для операции Allgather

1. *Начальный этап:* каждая ветвь p_i имеет блоки данных α_i
2-3. *Этап коммуникации:* на каждом шаге $k = 0, 1, \ldots, \lceil \log_2 p \rceil - 1$ ветвь p_i передаёт все *принятые* блоки данных ветви $\left(i - 2^k + P\right) \% P$ и принимает блоки данных от ветви $\left(i + 2^k\right) \% P$
4. *Завершающий этап:* каждая ветвь p_i выполняет циклический сдвиг всех принятых данных на *i* позиций



- Размер передаваемых данных удваивается на каждом шаге
- Количество обменов $2\lceil \log_2 P \rceil$

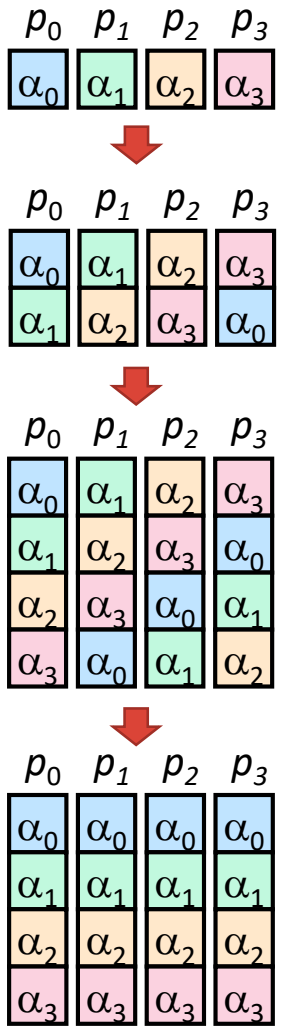
Bruck J., Ho C.-T., Kipnis S., Upfal E., Weathersby D. **Efficient algorithms for all-to-all communications in multiport message-passing systems** // IEEE Transactions on Parallel and Distributed Systems, 8(11):1143–1156, November 1997

Неблокирующая реализация алгоритма Дж. Брука в LibNBC

Псевдокод моей реализации алгоритма в Open MPI:

```
NBC_Sched_copy(sbuf, rbuf, scount)
tmpbuf = malloc((P – i) · rcount)
for dist = 1 to k do
    dest = (i – dist + P) % P
    src = (i + dist) % P
    if (dist ≤ (P ÷ 2)) then
        nblocks = dist
    else
        nblocks = P – dist
    end if
    NBC_Sched_send(rbuf, dest, nblocks · rcount)
    NBC_Sched_recv(rbuf, src, nblocks · rcount)    /* Ожидание */
    dist = dist · 2
end for
if (i ≠ 0) then
    NBC_Sched_copy(rbuf, tmpbuf, (P – i) · rcount)
    NBC_Sched_copy(rbuf + (P – i) · rcount, rbuf, i · rcount)
    NBC_Sched_copy(tmpbuf, rbuf + i · rcount, (P – i) · rcount)
end if
```

rbuf



- Количество обменов: $2\lceil \log_2 P \rceil$
- Операций копирования: 4
- Для выполнения сдвига необходим временный буфер
- Размер временного буфера в худшем случае: $O(P \cdot \alpha_i)$

Open MPI GitHub pull request: https://github.com/open-mpi/ompi/pull/6573

Организация экспериментов

- **Вычислительный кластер:**
  - 18 узлов: 2 x Intel Xeon E5420 (4 ядра; 2.5 ГГц), 8 Гб RAM
  - Сеть связи Gigabit Ethernet
  - Операционная система: Fedora Server 27 (ядро Linux 4.18.16-100.fc27.x86_64)
  - MPI: Open MPI 4.0 с моей реализацией алгоритма Дж. Брука для операции Iallgather в компоненте LibNBC

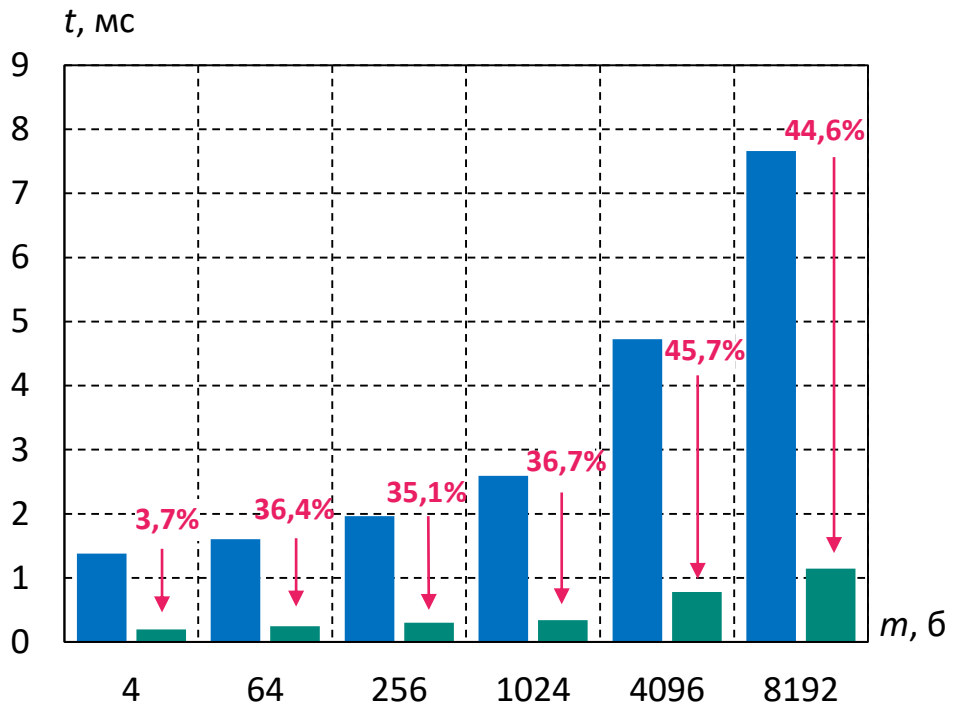
- **Тестовый пакет Intel MPI Benchmarks v2019.2:**
  - Набор для неблокирующих коллективных операций (IMB-NBC)
  - Набор для блокирующих коллективных операций (IMB-MPI1)
  - Число MPI-процессов $P = 24$

- **Показатели эффективности:**
  - Время t_{pure} выполнения коллективной операции Iallgather
  - Процент совмещения δ обмена с процессом вычисления за время t_{comp}:

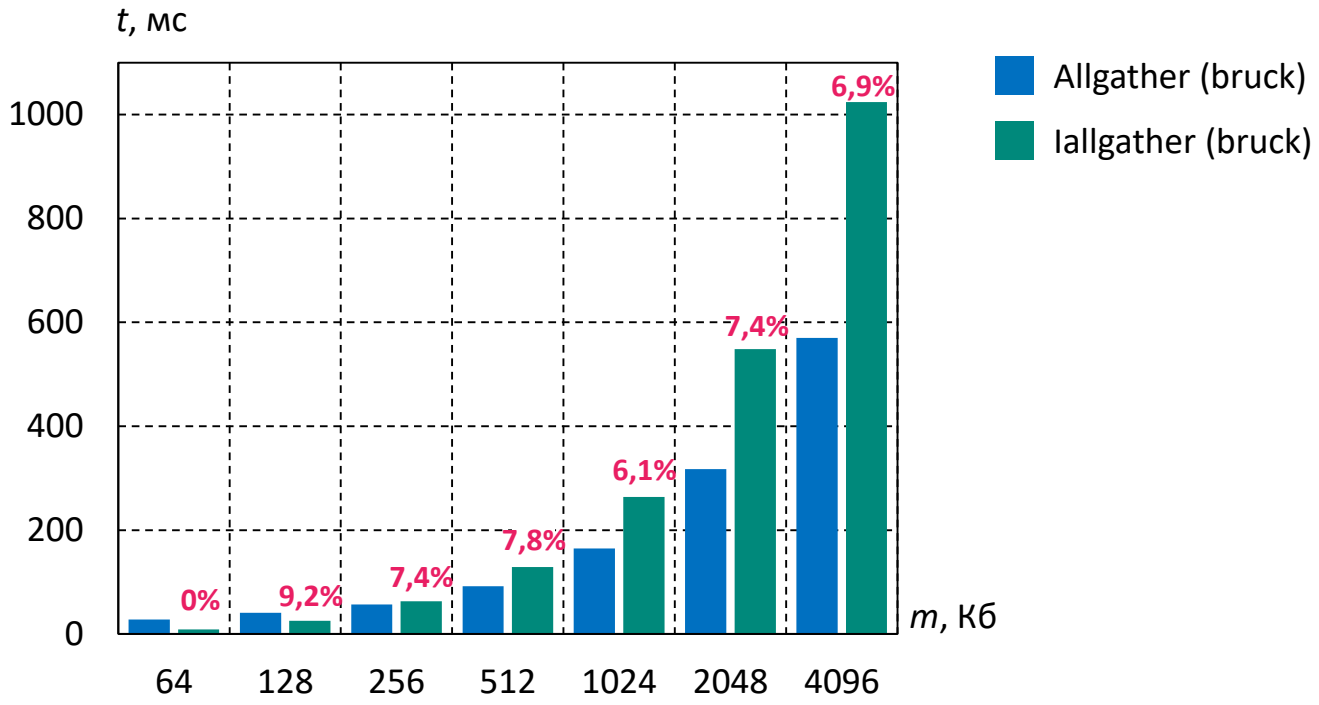
$$\delta = \max\left\{0, 100 - \frac{t_{total} - t_{comp}}{t_{pure}} \cdot 100\right\}$$

Результаты экспериментов

- На гистограммах ниже показана зависимость времени ***t*** выполнения (меньше лучше) блокирующей и неблокирующей версий коллективной операции Allgather от размера ***m*** данных для числа ветвей $P = 24$



а) $m = 4 \ldots 8192$ байт



б) $m = 64 \ldots 4096$ Кбайт

- **Метками** показан процент совмещения (больше лучше) времени обмена со временем вычислений для неблокирующей операции Iallgather
- Неблокирующая операция имеет меньшее время выполнения по сравнению с блокирующей
- Наибольший процент (45,7) совмещения был получен для данных малого размера

Заключение

- Выполнена реализация алгоритма Дж. Брука для неблокирующей коллективной операции Iallgather
- Сделан запрос на включение данной реализации в состав библиотеки Open MPI https://github.com/open-mpi/ompi/pull/6573
- Алгоритм Дж. Брука наиболее эффективен при небольших размерах данных
- Неблокирующая версия алгоритма имеет меньшее время выполнения по сравнению с блокирующей версией
- Наибольший процент (45,7) совмещения был получен для данных малого размера

Спасибо за внимание!

Аненков Александр Дмитриевич

Сибирский государственный университет телекоммуникаций и информатики

E-mail: anenkov.ru@gmail.com

Ежегодная российская научно-техническая конференция

«Обработка информации и математическое моделирование»

г. Новосибирск, 2019