УДК 519.6

МЕТОД ДЕКОМПОЗИЦИИ КОММУНИКАЦИОННЫХ ПРОСТРАНСТВ В РЕАЛИЗАЦИИ КОЛЛЕКТИВНЫХ ОПЕРАЦИЙ MPI

С. И. Сапронов
(ООО "Центр компетенций и обучения", г. Саров)

Рассматривается метод повышения производительности коллективных коммуникаций в конкретной реализации MPI за счет разбиения коммуникационных пространств, на которых изначально инициируются коллективные операции, на однородные подпространства и последующего выполнения этих операций по частям на отдельных подпространствах. При этом используются уже существующие алгоритмы коллективных операций.

Подход особенно эффективен на гетерогенных вычислительных платформах, использующих разнообразное коммуникационное оборудование.

Ключевые слова: MPI, *параллельные* вычисления, алгоритмы коллективных операций.

Введение

Современные высокопроизводительные вычислительные платформы чаще всего имеют неоднородную инфраструктуру, когда гетерогенные вычислительные узлы с многоядерной архитектурой соединены посредством разнообразного коммуникационного оборудования. Различные реализации библиотеки MPI (Message Passing Interface) [1] обеспечивают стандартизованный программный сервис для *параллельных* приложений, выполняемых на этих платформах. Коллективные операции, являясь наряду с двухточечными операциями одним из важнейших компонентов MPI, оказывают определяющее влияние на производительность параллельных приложений, которые их используют. При этом неоднородность вычислительных платформ может являться критическим фактором для эффективного использования этих операций. От способности той или иной реализации MPI учитывать неоднородные особенности вычислительной и коммуникационной среды напрямую зависит конкурентоспособность этой реализации. Таким образом, высокое качество и уровень реализации коллективных операций являются одними из основных целевых установок при разработке библиотек MPI.

Реализация коллективной операции обычно содержит некоторое количество алгоритмов, эффективность применения которых зависит от многих параметров: размеров передаваемых данных, вычислительных и коммуникационных характеристик, распределения процессов в вычислительной среде и др. Исполняющая среда в хорошей реализации MPI выбирает наиболее приемлемый (оптимальный) алгоритм для каждой коллективной операции в конкретных условиях применения или предоставляет пользователю гибкие средства управления выбором оптимального алгоритма. Разработка принципиально нового алгоритма, учитывающего изменяющиеся условия применения той или иной операции, — достаточно трудоемкий процесс.

В статье рассматривается альтернативный метод, позволяющий значительно увеличить количество используемых алгоритмов за счет применения уже существующих в реализации. В этом случае исполняющая среда MPI должна обеспечивать лишь поддержку механизма применения этого метода:

1) разбиения исходного коммуникационного пространства на некоторое количество непересекающихся регулярных подпространств и одного связующего подпространства;
2) разложения самой операции на составляю-

щие части, предназначенные для исполнения на каждом подпространстве;
3) выполнения операции на отдельных подпространствах по частям.

Этот подход особенно эффективно применим на гетерогенных вычислительных платформах, использующих разнообразное коммуникационное оборудование.

Метод характеризуется следующими положительными моментами:

- использованием достоинств в определенной среде исполнения уже имеющихся алгоритмов коллективных операций;
- одновременным независимым выполнением отдельных частей коллективной операции на непересекающихся коммуникационных подпространствах;
- возможным сокращением количества медленных коммуникаций при передаче данных.

Разбиение исходного коммуникационного пространства

Каждая операция обмена в MPI выполняется под управлением специального объекта — коммуникатора, который объединяет определенную совокупность MPI-процессов и обеспечивает независимый контекст выполнения операции. В коллективной операции участвуют все процессы, выполняемые под управлением коммуникатора. Будем использовать для их обозначения термин *коммуникационное пространство*.

Разбиение исходного коммуникационного пространства на непересекающиеся (регулярные) подпространства осуществляется при первом использовании соответствующего коммуникатора в какой-либо коллективной операции (рис. 1).

Помимо регулярных подпространств, покрывающих исходное коммуникационное пространство, создается дополнительное подпространство, которое служит для связи независимых подпространств между собой. Это *связующее* подпространство содержит ровно по одному процессу из каждого регулярного подпространства. Процессы из связующего подпространства назовем главными процессами регулярных подпространств.

Критерием для разбиения может быть любое отношение эквивалентности на множестве MPI-процессов, составляющих группу исходного коммуникатора. Приведем несколько примеров.

1. Классическое отношение эквивалентности — это расположение процессов на одном вычислительном узле кластера. В данном случае каждое подпространство будет содержать только расположенные на этом конкретном узле процессы. При этом для внутренних коммуникаций на узле могут быть задействованы коллективные алгоритмы, использующие быстрые обмены через сегменты общей памяти.
2. Выделение в отдельные подпространства процессов, соединенных посредством специфического оборудования, например FCA (Fabric Collective Accelerator) [2]. Внутри таких подпространств доступны все преимущества FCA-обменов.
3. Явное перечисление элементов каждого подмножества. Такой критерий может быть использован, когда известна специфика организации обменов в приложении и пользователь может задать наиболее подходящее разбиение.

Исполнительная среда MPI предоставляет пользователю средства управления выбором того или иного критерия разбиения исходных коммуникационных пространств посредством специальных переменных среды и/или опций командной строки запуска. При этом автоматически определяется состав и топология вычислительной платформы и осуществляется селекция оптимальных алгоритмов коллективных операций для конкретных условий применения.

Разложение коллективных операций

Выбор того или иного способа разложения каждой коллективной операции на составные части базируется на организации обработки потоков данных в этой операции. По этому критерию выделяются следующие пять групп операций: *один — всем*, *все — одному*, *все — всем*, *барьерная синхронизация*, *частичная редукция*. В каждой группе действуют свои правила разложения операции на независимые части, которые предназначены для выполнения на отдельных подпространствах.

Один — всем. Операции этой группы характеризуются наличием одного выделенного MPI-процесса, который определенным способом распространяет данные среди всех прочих процессов. Выделенный процесс называется корневым.

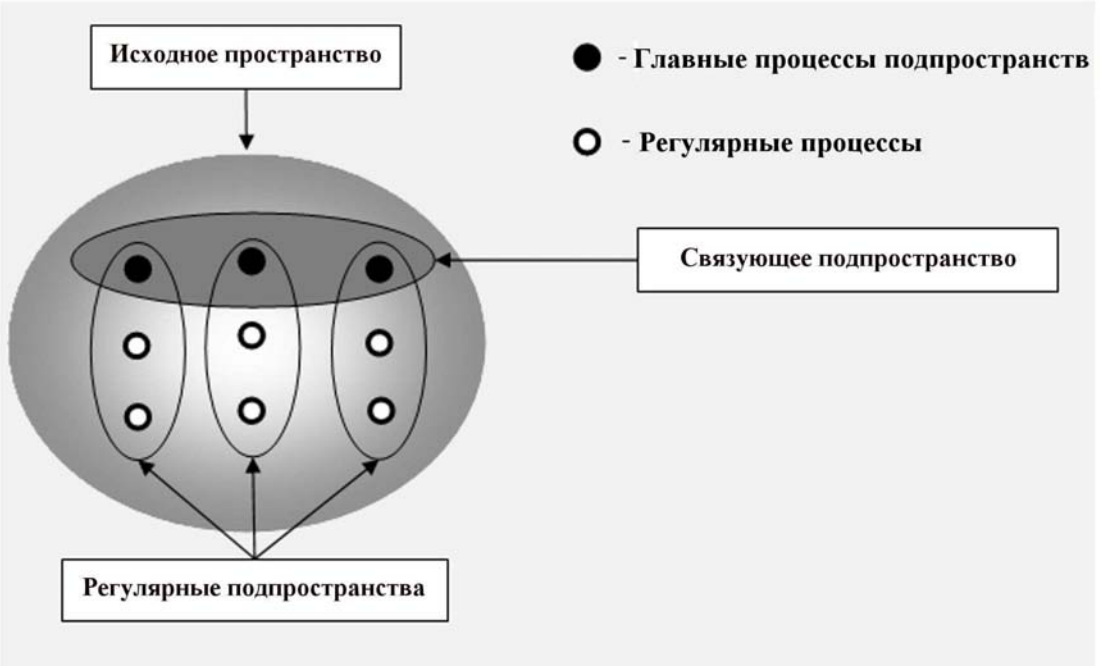

Рис. 1. Декомпозиция коммуникационного пространства

В группу входят следующие операции: *bcast*, *scatter*, *scatterv*.

Схематически разложение этих операций представлено на рис. 2. Операции выполняются за два шага. На первом шаге корневой процесс рассылает данные главным процессам в контексте пространства связи. На втором шаге каждый главный процесс рассылает данные в контексте своего регулярного подпространства, выступая в роли корневого процесса уже для локальной операции на этом подпространстве. На втором шаге операции выполняются одновременно — каждая на своем независимом подпространстве.

Все — одному. В операциях этой группы также выделен корневой процесс. Однако в отличие от группы *один — всем* здесь он не распространяет, а собирает данные от остальных процессов.

К этой группе относятся следующие операции: *reduce*, *gather*, *gatherv*.

Схема выполнения этих операций отличается от схемы *один — всем* направлением передачи данных и очередностью шагов. Операции выполняются за два шага. Первый шаг заключается в одновременном выполнении операций *reduce/gather* на всех регулярных подпространствах, где корневыми процессами являются главные процессы каждого регулярного подпространства. На втором шаге результат аккумулируется целевым корневым процессом в итоге выполнения операции *reduce/gather* на связующем подпространстве.

Все — всем. Это самая большая группа коллективных операций. Здесь все процессы участвуют как в сборе, так и распространении данных.

Выделяются две подгруппы:

1) *allreduce*, *reduce_scatter*, *allgather*, *allgatherv*;
2) *alltoall*, *alltoallv*, *alltoallw*.

Каждую операцию первой подгруппы можно представить в виде композиции двух коллективных операций — базовой корневой и операции распространения:

$allreduce = reduce + bcast$;
$reduce\_scatter = reduce + scatterv$;
$allgather = gather + bcast$;
$allgatherv = gatherv + bcast$.

В качестве корневого процесса для определенности выбирается нулевой процесс. Далее каждая из композиционных операций разлагается согласно схемам *все — одному* или *один — всем*, описанным выше. В итоге для операций первой подгруппы получаем четырехшаговую схему, представленную на рис. 3. На первых двух шагах выполняется соответствующая операция *все — одному* (*reduce, gather, gatherv*) с нулевым

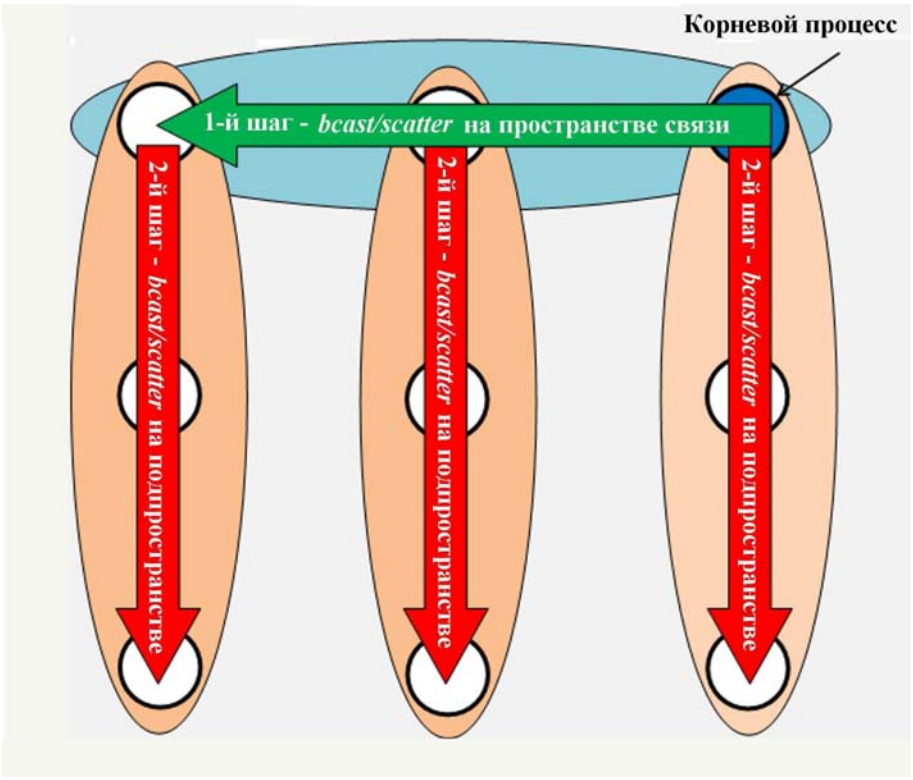

Рис. 2. Выполнение операции *один — всем* по частям

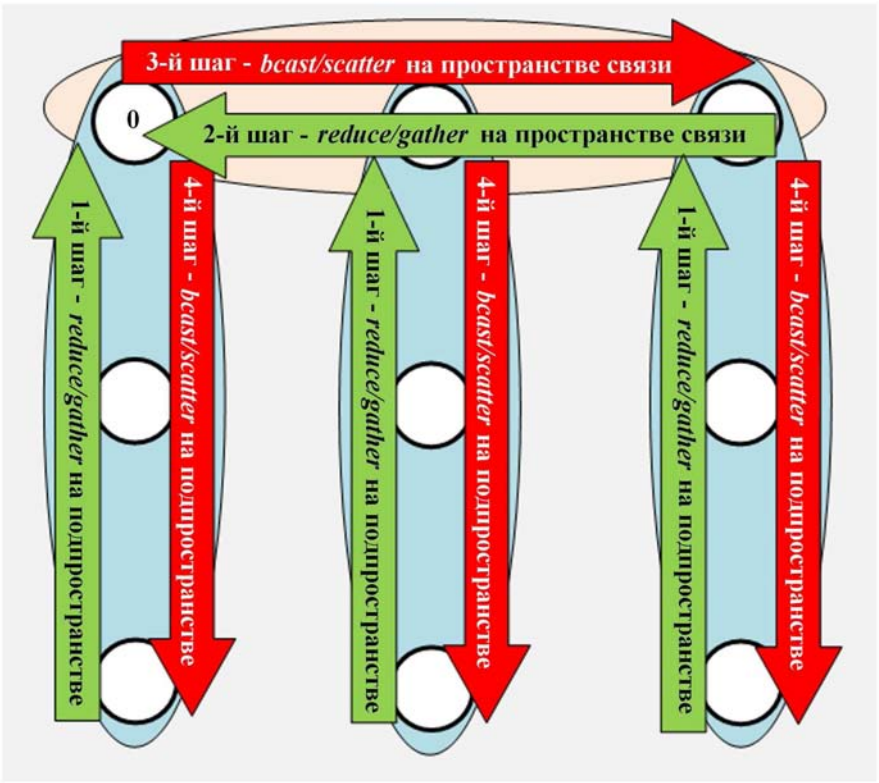

Рис. 3. Выполнение операции *все — всем* первой подгруппы по частям

процессом в качестве корневого. На последних двух шагах данные, полученные нулевым процессом, рассылаются всем остальным процессам согласно семантике операций этой группы.

Операции семейства *alltoall* выполняются по следующей схеме (рис. 4):

1) сбор главными процессами каждого регулярного подпространства данных, предназначенных для распространения вне этих подпространств, посредством операции *gatherv*;
2) выполнение локальных операций *alltoall* на регулярных подпространствах для данных, которые предназначены для распространения внутри каждого из этих подпространств;
3) рассылка главным процессам регулярных подпространств данных, предназначенных для каждого из этих подпространств, посредством операции *alltoall* на связующем подпространстве;
4) выполнение на каждом регулярном подпространстве операции *scatterv*, которая рассылает собранные главным процессом данные всем процессам регулярного подпространства.

Барьерная синхронизация. В этой группе представлена единственная коллективная операция *barrier*, которая выполняется за три шага (рис. 5):

1) одновременная операция *barrier* на всех регулярных подпространствах;
2) *barrier* на связующем подпространстве;
3) *bcast* на каждом регулярном подпространстве с главным процессом в качестве корня.

Частичная редукция. Группа включает две операции: *scan* и *exscan*. Здесь каждый процесс с рангом (rank) R получает свои индивидуальные данные в результате выполнения операции редукции входных данных для процессов с рангами $0, \ldots, R$ *(scan)* или $0, \ldots, R-1$ *(exscan)*.

Схема выполнения операций представлена на рис. 6. Сначала входные данные от всех процессов собираются нулевым процессом и их копии передаются всем главным процессам, где выполняется операция редукции индивидуально для каждого процесса. На последнем шаге результаты рассылаются процессам-адресатам, на каждом регулярном пространстве независимо от других.

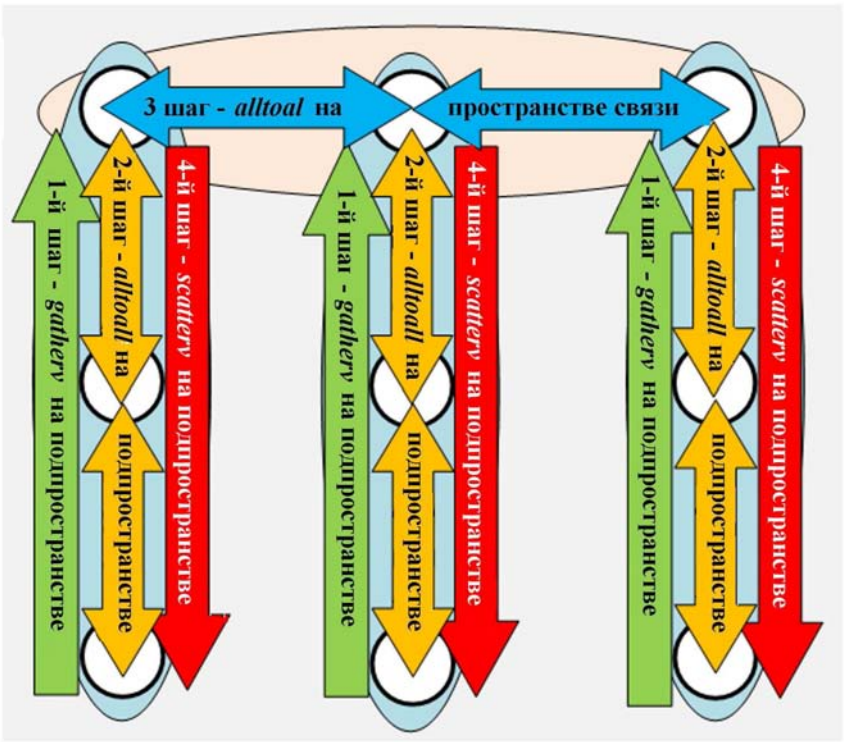

Рис. 4. Выполнение операций семейства *alltoall* по частям

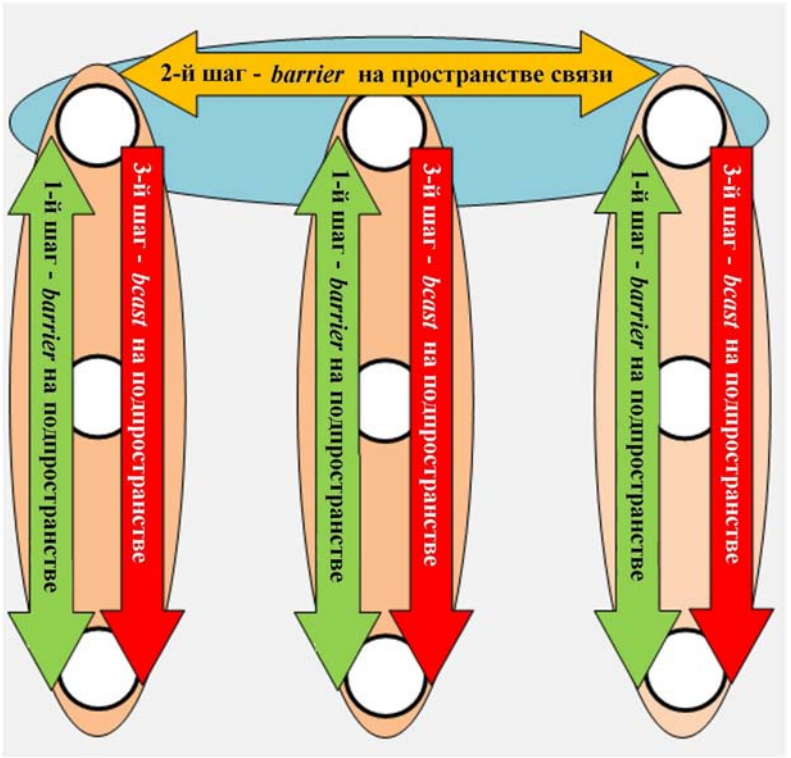

Рис. 5. Выполнение барьерной синхронизации по частям

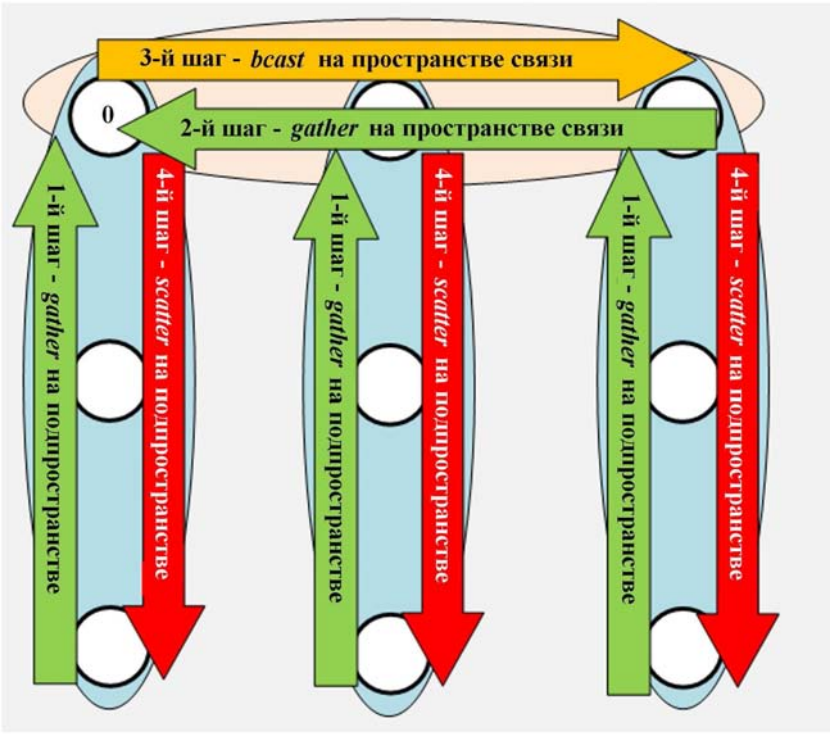

Рис. 6. Выполнение операции частичной редукции по частям

Пример использования метода декомпозиции

Рассмотрим применение метода декомпозиции на конкретном примере:

- кластер на базе Intel® Xeon™ X5560 (два 4-ядерных узла);
- коммуникации: shared memory внутри узлов, 1Gig Ethernet между узлами;
- загрузка приложения: round robin;
- приложение: тест *bcast* для восьми процессов из набора IMB (Intel MPI Benchmark) [4];
- биноминальный алгоритм.

На рис. 7 представлена общая схема биноминального алгоритма для восьми процессов, нулевой процесс распространяет данные среди остальных. Алгоритм рекурсивный по группам процессов. Начальная группа — все процессы.

Шаг рекурсии:

- группа процессов делится пополам на две подгруппы (левую и правую на рис. 7);
- процесс с минимальным рангом из левой группы пересылает данные процессу с минимальным рангом из правой группы.

За три шага все процессы получат распространяемые данные.

На рис. 8 представлены две схемы выполнения операции *bcast* — с использованием и без использования метода декомпозиции.

При декомпозиции операция *bcast* выполняется следующим образом (передачи обозначены светлыми стрелками):

1) *bcast* на связующем пространстве. Главный процесс первого узла, являющийся корневым, пересылает данные главному процессу второго узла;
2) *bcast* на каждом узле. Главный процесс распространяет данные среди процессов узла по биноминальной схеме.

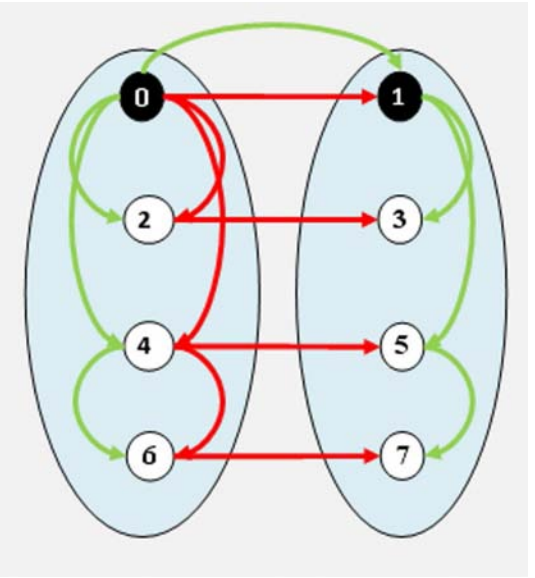

Рис. 8. Две схемы выполнения операции *bcast*: ⟶ — с декомпозицией; ⟶ — без декомпозиции

Передачи в классическом случае на рис. 8 обозначены темными стрелками.

Видно, что в случае декомпозиции выполняется только одна медленная передача — это первая передача с узла на узел. Остальные передачи быстрые, они выполняются через общую память одновременно на обоих узлах. Без декомпозиции выполняются четыре медленные передачи между узлами и три быстрые. Временные результаты для этих вариантов представлены на рис. 9 в виде диаграммы, где вертикальная ось — шкала длин сообщений от 1 байта до 4 Мбайт; на горизонтальной оси — отношение времени выполнения операции с декомпозицией ко времени выполнения без декомпозиции.

Для рассмотренного примера метод декомпозиции позволяет ускорить операцию *bcast* от 2,5 раз (для длинных сообщений) до 5 раз (для коротких сообщений).

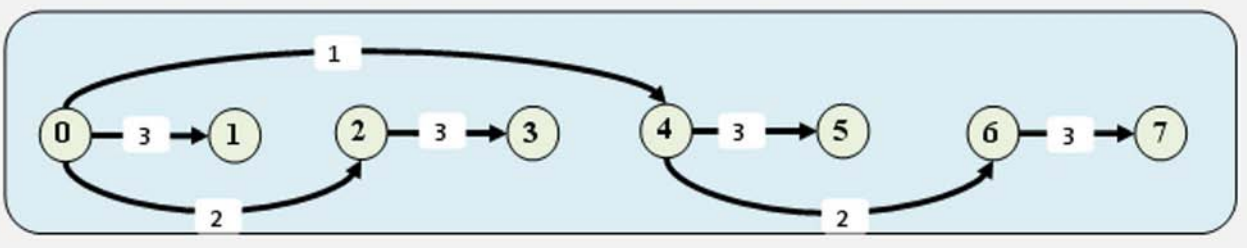

Рис. 7. Схема биноминального алгоритма *bcast*

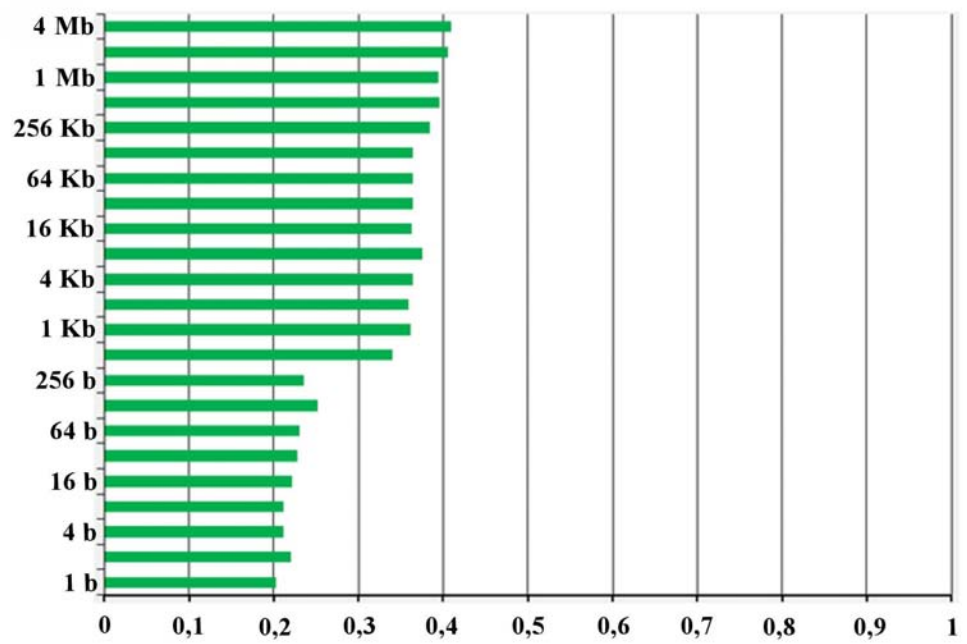

Рис. 9. Сравнительные результаты выполнения операции *bcast* с декомпозицией и без декомпозиции

Заключение

Исследование метода декомпозиции было выполнено автором на базе *открытого* кода Open MPI [3], компонентная организация которого значительно облегчила работу.

Планируется продолжение исследований в рамках Open MPI в направлении расширения и совершенствования возможностей доступа параллельных приложений, использующих MPI, к инфраструктуре многоядерных платформ с целью максимально эффективного использования новых аппаратных и программных средств. Предполагается обратить особое внимание на FCA [2]. Эти средства, обеспечивающие поддержку коллективных операций на аппаратном уровне, имеют обнадеживающие предпосылки для успешного развития.

Работа выполнена в рамках контракта (№ 07.524.12.4020) с Министерством образования и науки РФ.

Список литературы

1. *Message Passing Interface Forum.* MPI: A Message Passing Interface // Proc. of "Supercomputing'93". Los Alamitos: IEEE Computer Society Press, 1993. P. 878—883.
2. Fabric Collective Accelerator (FCA). http://www.mellanox.com/related-docs/prod_acceleration_software/FCA.pdf
3. *Gabriel E., Fagg G. E., Bosilca G. et al.* Open MPI: goals, concept, and design of a next generation MPI implementation. http://www.open-mpi.org/papers/euro-pvmmpi-2004-overview/euro-pvmmpi-2004-overview.pdf
4. Intel MPI Benchmark. http://www.uybhm.itu.edu.tr/ documents/IMB_ug-3.0.pdf

Статья поступила в редакцию 29.08.12.