Феоктистов Александр Геннадьевич – Институт динамики систем и теории управления им. В.М. Матросова СО РАН; e-mail: agf65@yandex.ru; 664033, г. Иркутск, ул. Лермонтова, 134; тел.: +79247116704; с.н.с.; к.т.н.; доцент.

Костромин Роман Олегович – e-mail: romang70055@gmail.com; тел.: +79041150109; аспирант.

Feoktistov Aleksandr Gennadyevich – Matrosov Institute for System Dynamics and Control Theory of SB RAS; e-mail: agf65@yandex.ru; 134, Lermontov street, Irkutsk, Russia; phone: +79247116704; senior research officer; cand. of eng. sc.; associate professor.

Kostromin Roman Olegovich – e-mail: romang70055@gmail.com; phone: +79041150109; post-graduate student.

УДК 004.272 DOI 10.18522/2311-3103-2016-11-7587

М.Г. Курносов

АНАЛИЗ МАСШТАБИРУЕМОСТИ АЛГОРИТМОВ КОЛЛЕКТИВНЫХ ОБМЕНОВ НА РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ*

При разработке параллельных программ для вычислительных систем (ВС) с массовым параллелизмом значительное место по частоте использования и приходящемуся на них суммарному времени выполнения занимают коллективные операции обменов информацией (групповые, глобальные, collective communications). В них участвуют все ветви параллельной программы: трансляционная передача («один-всем», one-to-all broadcast/scatter), коллекторный прием («все-одному», all-to-one gather/reduce), трансляционно-циклический обмен («каждый-всем», all-to-all gather/reduce). Для реализации каждой коллективной операции, имеется множество алгоритмов, из которого необходимо выбрать оптимальный – обеспечивающий минимум времени выполнения операции. Для такого сравнительного анализа в моделях параллельных вычислений (BSP – bulk synchronous parallel, Дж. Хокни, LogP, LogGP, PLogP) строят аналитические оценки времени выполнения алгоритмов, как функции от параметров системы и коллективной операции: числа процессоров, показателей производительности каналов связи, размеров передаваемых сообщений. В данной работе для коллективной операции корневой редукции (all-to-one reduce) в модели параллельных вычислений LogP построены аналитические выражения (оценки) времени выполнения алгоритмов ее реализации. В отличие от известных работ, выражения построены для общих и частных (особых) случаев значений параметров вычислительной системы и коллективных операций. Для учета копирования сообщений в памяти вычислительных узлов модель LogP расширена дополнительным параметром λ – время, требуемое на копирование одного байта в памяти вычислительного узла. На примере алгоритма -параллельных цепочек продемонстрирован подход к построению оптимальных в модели LogP алгоритмов коллективных операций. Предложенный оптимизированный алгоритм -параллельных цепочек реализован в стандарте MPI. Результаты экспериментов на вычислительных кластерах с сетями связи стандарта MPI подтверждают полученные теоретические результаты, в частности рекомендации относительно выбор числа k цепочек. Выбор конкретной математической модели параллельных вычислений обусловлен спецификой алгоритма и целевой ВС. Например, если алгоритм реализует группировку сообщений в пакеты больших размеров, то целесообразно использовать модель LogGP, которая в явном виде учитывает издержки на передачу сообщений больших размеров.

Коллективные обмены; глобальные обмены; LogP; MPI; параллельное программирование; вычислительные системы.

* Работа выполнена при поддержке РФФИ (проекты 15-37-20113, 15-07-00653).

M.G. Kurnosov

SCALABILITY ANALYSIS OF COLLECTIVE OPERATIONS BASED ON LOGP MODEL

Performance of collective communications (one-to-all broadcast/scatter, all-to-one gather/reduce, all-to-all gather) is critical for high-performance large-scale computer systems and parallel algorithms. In this paper, analytical expressions in LogP model for execution time of collective operations are proposed. In contrast to the well-known results, the proposed expressions are constructed for general and special cases of parameters and values of the computer system and collective operation (number of processes, selection of root process). Such estimates are needed for the analysis of algorithms scalability on large-scale computer systems. Expressions are obtained as functions of process number. This makes it possible to analyze the imbalance of process loading. The LogP model is extended by parameter λ – a time for transferring a one byte message over shared memory. Approach for building optimal algorithms in LogP model is demonstrated on example of k-chain algorithm. For this algorithm the optimal value of k in the LogP model is found. A new algorithm based on the optimal value of k is developed. The dependence of the execution time of the proposed algorithm on the number of processes has a growth rate of $O(\sqrt{P})$, which is more efficient compared to the linear running time of the original k-chain algorithm. The proposed algorithms are implemented in the MPI standard and studied on computer clusters with InfiniBand QDR networks. The choice of the model of parallel computation depends on the specifics of the algorithm and the target computer system. For example, if the algorithm realizes the groups of messages in larger packages, it is advisable to use the LogGP model.

Collective operations; reduce; LogP; MPI; parallel programming; message passing.

Введение. Распределенный характер памяти вычислительных систем (ВС) с массовым параллелизмом определяет базовым механизмом информационных взаимодействий процессов параллельных программ передачу сообщений по каналам межмашинных связей (message-passing). Среди основных схем информационных обменов значительное место по частоте использования и приходящемуся на них суммарному времени выполнения занимают коллективные операции обмена информацией (групповые, глобальные, collective communications) [1, 2].

Такие операции делятся на корневые и некорневые. В них участвуют все ветви параллельной программы. К корневым относятся:

- трансляционная передача («один-всем», one-to-all broadcast/scatter);
- коллекторный прием («все-одному», all-to-one gather/reduce).

Примерами некорневого обмена служат обмены типа «каждый-всем» (all-to-all gather/reduce).

Для широкого класса параллельных алгоритмов время выполнения коллективных операций является критически важным и определяет их масштабируемость [3, 4]. Работы, ориентированные на оптимизацию коллективных обменов в ВС, ведутся по двум направлениям.

1. Создание структурно-ориентированных алгоритмов реализации коллективных операций для систем с фиксированной топологией сети межмашинных связей: тороидальной (Cray Gemini, IBM PERCS, Fujitsu Tofu, SKIF 3D-torus) [5–7], гиперкубической (СМПО, Ангара) [2, 9], древовидной и кольцевой (InfiniBand fat tree, ТН Express-2, МВС-Экспресс) [8] и др. В рамках данного подхода разработчикам известны топология системы, алгоритмы маршрутизации сообщений, зависимость производительности виртуальных каналов связи от размеров передаваемых сообщений, а также методы выделения подсистемы для задач пользователей. Наличие такой информации о целевой системе позволяет создавать оптимальные для нее алгоритмы реализации коллективных обменов.

2. Разработка масштабируемых алгоритмов реализации коллективных операций на базе примитивов двухсторонних обменов (send, recv) в отрыве от конкретной топологии коммуникационной сети. Такой подход реализуется переносимыми системами параллельного программирования, ориентированными на широкий спектр архитектур вычислительных систем. В их числе коммерческие и свободно распространяемые библиотеки стандарта MPI (MPICH/MVAPICH, Open MPI, Intel MPI) и GASNet. Очевидно, что игнорирование такими алгоритмами ряда структурных характеристик системы не позволяет им достигать предельных возможностей коммуникационной сети ВС. Однако, на практике такой баланс между узкой специализацией алгоритмов на фиксированной топологии и их переносимостью между широким спектром архитектур ВС обеспечивают эффективность близкую к оптимальной [3, 4, 10, 17].

В данной работе внимание уделено второму подходу – анализу и оптимизации алгоритмов коллективных обменов на базе примитивов двусторонних обменов.

Основные схемы коллективных обменов реализуются алгоритмами рассылки данных по кольцу (ring), рекурсивного удваивания (recursive doubling), рекурсивного деления пополам (recursive halving), алгоритмом Брука (J. Bruck), попарных обменов (pairwise exchange) и алгоритмами, упорядочивающими процессы в деревья различных видов (биномиальные деревья, сбалансированные *k*-арные деревья, плоские деревья, конвейеры/цепочки) [10–12]. Таким образом, для реализации каждой коллективной операции, имеется множество алгоритмов, из которого необходимо выбрать оптимальный – обеспечивающий минимум времени выполнения операции. Для такого сравнительного анализа в моделях параллельных вычислений (BSP – bulk synchronous parallel, Дж. Хокни, LogP, LogGP, PLogP) строят аналитические оценки времени выполнения алгоритмов, как функции от параметров системы и коллективной операции: числа процессоров, показателей производительности каналов связи, размеров передаваемых сообщений [2, 13, 14]. Аналитические выражения такого рода позволяют решать две важные задачи:

1. Оценивать масштабируемость алгоритмов на существующих и перспективных системах – оценивать скорость роста функции, отражающей зависимость времени выполнения алгоритма от числа процессоров в ВС;

2. Проводить сравнительный анализ времени выполнения алгоритмов на заданной подсистеме ВС для заданных параметров коллективной операции (включая предсказательное моделирование).

В известных работах [2, 3, 10, 14, 15] в моделях Дж. Хокни, LogP, LogGP, PLogP предложены аналитические оценки времени выполнения алгоритмов коллективных обменов как интегральные выражения для всего алгоритма (оценка по максимальному времени выполнения всех процессов). Однако для анализа равномерности загрузки процессоров востребованы выражения в виде функций от номера процесса. Кроме этого, в аналитических оценках времени выполнения алгоритмов коллективных обменов необходимо учитывать «особые случаи», при которых алгоритмы имеют ограничения по применимости или дополнительные накладные расходы. Например: число процессов является (не является) степенью числа 2; в качестве корневого процесса выбран процесс отличный от 0; заданная бинарная операция в редукции не является коммутативной и пр.

В данной работе в модели параллельных вычислений LogP предложены аналитические выражения (оценки) времени выполнения алгоритмов как функции от параметров ВС и коллективной операции. Выражения построены для общих и частных (особых) случаев значений параметров ВС и коллективных операций. Для учета копирования сообщений в памяти вычислительных узлов модель LogP расширена дополнительным параметром λ – время, требуемое на копирование одного

байта в памяти вычислительного узла. Построенные аналитические выражения и подходы к оптимизации в модели LogP алгоритмов коллективных операций проиллюстрированы на примере корневой редукции (all-to-one reduce) стандарта MPI.

1. Операция корневой редукции. Операция корневой редукции *MPI_Reduce* комбинирует при помощи заданной бинарной ассоциативной операции *op* элементы буферов отправки *sbuf* всех процессов и записывает результат в буфер приема *rbuf* корневого процесса *root*.

```
MPI_Reduce(sbuf, rbuf, count, datatype, op, root, comm)
```

Функция вызывается всеми процессами группы коммуникатора *comm*. Каждый процесс передает в функцию адрес буфера передачи *sbuf*, содержащий *count* элементов типа *datatype*. Его содержимое после выполнения операции остается без изменения. Корневой процесс *root* записывает результат операции в свой локальный буфер приема *rbuf*, также содержащий *count* элементов.

2. Модель параллельных вычислений LogP. Модель LogP – это математическая модель многопроцессорной ВС с распределенной памятью, в которой параллельные процессы взаимодействуют посредством двусторонних обменов короткими сообщениями фиксированного размера [16, 17]. Модель игнорирует структуру коммуникационной сети ВС и отражает лишь показатели производительности каналов связи между процессорами.

Основные параметры модели:

L – верхняя граница латентности (latency, delay) передачи сообщения одного процессора другому;

o – промежуток времени (overhead), в течение которого процессор занят передачей или приемом сообщения (в течение этого времени процессор не может выполнять другие операции);

g – минимальный интервал времени (gap) между последовательными передачами или приемами сообщений ($1/g$ – пропускная способность канала связи, доступная процессору);

P – количество процессоров в системе.

Параметры L, o и g измеряются в тактах работы процессора, но могут быть выражены в секундах. Неявным параметром модели является размер w сообщения. Предполагается, что сообщения имеют небольшой размер, одно или несколько машинных слов.

В модели LogP время t передачи сообщения от одного процессора другому выражается как

$$t = 2o + L.$$

На рис. 1. показан пример последовательной передачи процессором 0 трех сообщений процессору 1. В нулевой момент времени процессор 0 инициирует передачу сообщения процессору 1. Для этого ему требуется o единиц времени на выдачу сообщения в сеть, далее, через L единиц времени сообщение достигает процессора 1. Процессор 1 затрачивает o единиц времени для получения сообщения из сети. Далее процессор 0 отправляет второе сообщение. Он не может начать передачу менее, чем через $\max\{o, g\}$ единиц времени с момента начала передачи первого сообщения. В приведенном примере предполагается, что $g > o$. Таким образом, вторая и третья передачи сообщений процессором 0 начинаются в моменты времени g и $2g$ соответственно. Процессор 0 завершает работу в момент времени $2g + o$, а процессор 1 завершает прием данных за время $L + 2g + 2o$.

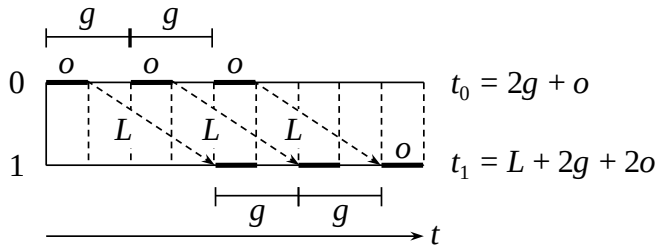

Рис. 1. Пространственно-временная диаграмма взаимодействия процессоров в модели LogP: процессор 0 выполняет три передачи сообщения процессору 1 ($g > o$)

Модель основана на ряде допущений. Все процессоры функционируют асинхронно и, как следствие, латентность передачи сообщений может варьироваться вплоть до значения L. Недетерминированный характер латентности может привести к тому, что сообщения, предназначенные для одного процессора, могут быть доставлены в порядке, отличном от исходного.

3. Алгоритм биномиального дерева для корневой редукции. В алгоритме биномиального дерева (binomial tree) процессы логически выстраиваются в дерево обменов, представляющее собой совокупность биномиальных деревьев, степени которых определяются номерами значащих битов в двоичном представлении числа P процессов, участвующих в операции корневой редукции. На рис. 2 приведен пример дерева для $P = 22_{10} = 10110_2$ – деревья B_4, B_2 и B_1.

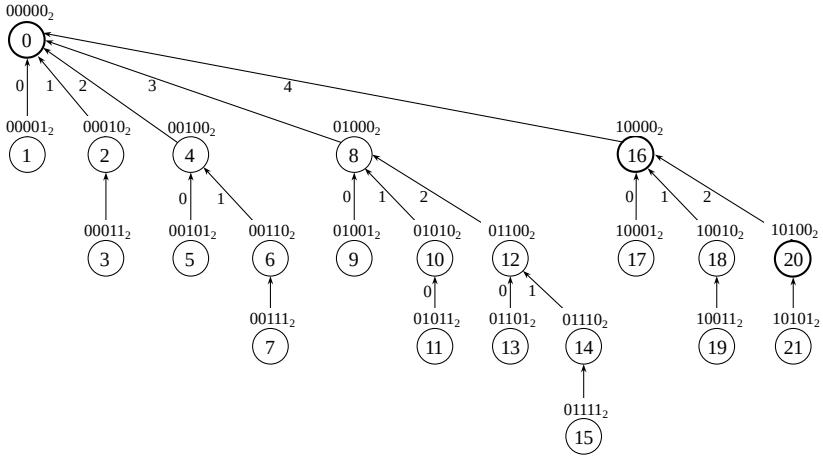

Рис. 2. Дерево обменов для $P = 2210 = 101102$: деревья $B4$, $B2$ и $B1$ (стрелками показаны направления передачи данных, номер дуги – шаг, на котором выполняется передача, узлам 16 и 20 не хватает потомков)

Каждый процесс $rank \in \{0, 1, \ldots, P - 1\}$ выполняет цикл из $\lceil \log_2 P \rceil$ раундов обмена данными. На шаге $i = 0, 1, \ldots, \lceil \log_2 P \rceil - 1$ процесс передает данные либо их принимает:

- если бит i в номере $rank$ процесса установлен, то процесс передает данные процессу $peer$, в номере которого этот бит сброшен $peer = rank \,\&\, (\sim (1 \ll i))$;

- если бит i в номере $rank$ процесса не установлен, то процесс принимает данные от процесса $peer$, в номере которого этот бит установлен $peer = rank \mid (1 \ll i)$.

В данной работе построены оценки времени выполнения алгоритма для следующих важных случаев параметров ВС и операции.

3.1. Случай коммутативной бинарной операции редукции. Если бинарная операция *op* является коммутативной (таковыми являются все операции, предопределенные в стандарте MPI), то локальные редукции в процессах могут выполняться в произвольном порядке.

Случай 1. Число процессов является степенью числа 2. На рис. 3 приведена пространственно-временная диаграмма в модели LogP выполнения алгоритма биномиального дерева для коммутативной операции и $P = 16$.

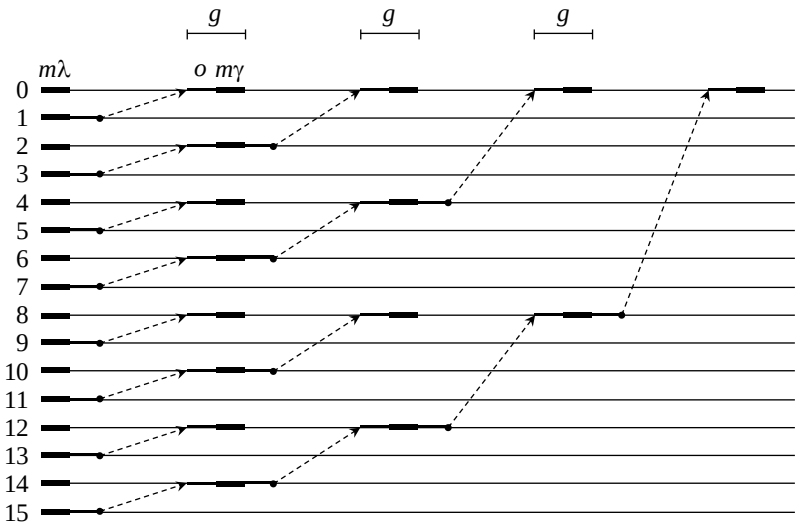

Рис. 3. Пространственно-временная диаграмма в модели LogP выполнения алгоритма биномиального дерева для коммутативной операции: $P = 16$, $root = 0$

Время выполнения алгоритма определяется временем выполнения корневого процесса, который копирует свой буфер $sbuf$ в буфер $rbuf$ и выполняет $\log_2 P$ операций приема данных, и столько же локальных редукций. Время его выполнения в модели LogP равно

$$t_{root}(m) = m\lambda + (\log_2 P - 1)(o + L + \max\{o + m\gamma, g\}) + 2o + L + m\gamma,$$

где $m\lambda$ – издержки на копирование буфера $sbuf$, а $m\gamma$ – временные затраты на выполнения локальной редукции процессором.

Каждый листовой процесс копирует буфер $sbuf$ в $rbuf$, выполняет передачу и завершает свою работу. Отсюда время его выполнения:

$$t_{leaf}(m) = m\lambda + o.$$

Внутренний процесс с номером r выполняет $\mathrm{CTZ}(r)$ операций приема данных, где $\mathrm{CTZ}(r)$ – число завершающих нулевых битов (count trailing zeros) в двоичном представлении числа r. Время выполнения внутреннего процесса r включает издержки на копирование буфера передачи в $rbuf$, время выполнения обменов и локальных редукций:

$$t_r(m) = m\lambda + (\mathrm{CTZ}(r) - 1)(o + L + \max\{o + m\gamma, g\}) + 3o + L + m\gamma.$$

Случай 2. Число процессов не является степенью числа 2. В этом случае (рис. 2) время работы алгоритма также определяется временем корневого процесса

$$t_{root}(m) = m\lambda + \lfloor \log_2 P \rfloor (o + L + \max\{o + m\gamma, g\}) + o + m\gamma.$$

Время выполнения листовых процессов не отличается от выражения, полученного при рассмотрении предыдущего случая. Однако, некоторые из внутренних процессов выполняют меньше CTZ(r) операций приема данных. Например, на рис. 2 процессам 16 и 20 не хватает потомков для взаимодействий. Формально, внутреннему узлу r не хватает потомков, если для него выполняется неравенство: $r + 2^{\mathrm{CTZ}(r)-1} > P - 1$.

Из построенных аналитических выражений видно, что время выполнения алгоритма логарифмически зависит от числа P процессов, что свидетельствует о его хорошей масштабируемости. Вычислительная нагрузка в алгоритме распределена по процессам неравномерно на каждом шаге число процессов, выполняющих локальные редукции сокращается в два раза. Последнее, на практике, ограничивает эффективность данного алгоритма для сообщений значительных размеров: $m \geq cP$, c = const. Для сообщений больших размеров целесообразно использовать алгоритм Р. Рабенсейфнера, который вовлекает в выполнение локальных редукций все процессы [10].

3.2. Случай некоммутативной операции редукции. В данной вариации алгоритма биномиального дерева учитывается некоммутативность операции *op* локальной редукции. Комбинирование данных выполняется с учетом номеров процессов (последовательного порядка выполнения редукций). В каждом внутреннем процессе $rank$ правым операндом операции *op* всегда являются данные, полученные от процесса с номером $rank + 1$.

Для этого процессы реализуют обмены, считая, что корневым является процесс с номером $rank = 0$. Структура обменов в основном цикле идентична обменам в алгоритме для случая коммутативной операции. После цикла обменов процесс 0 пересылает результат корневому процессу с номером $root$.

Случай 1. Корневым является процесс с номером 0. Если количество P процессов является степенью числа 2, то время выполнения алгоритма равно

$$t_{root}(m) = 2m\lambda + (\log_2 P - 1)(o + L + \max\{o + m\gamma + m\lambda, g\}) + 2o + L + m\gamma.$$

Если же количество процессов не является степенью двойки, то время работы алгоритма есть

$$t_{root}(m) = 2m\lambda + \lfloor \log_2 P \rfloor (o + L + \max\{o + m\gamma + m\lambda, g\}) + o + L + m\gamma.$$

Случай 2. Корневым является процесс с положительным номером. В этом случае время выполнения корневого процесса с номером $root > 0$ складывается из времени выполнения процесса 0 и времени, требуемого на передачу результата операции от процесса 0 процессу $root$. Здесь аналогично, если количество процессов – степень числа 2, то время работы корневого процесса

$$t_{root}(m) = 2m\lambda + (\log_2 P - 1)(o + L + \max\{o + m\gamma + m\lambda, g\}) + 4o + 2L + m\gamma.$$

Если же количество процессов не является степенью двойки, то время работы алгоритма

$$t_{root}(m) = 2m\lambda + \lfloor \log_2 P \rfloor (o + L + \max\{o + m\gamma + m\lambda, g\}) + 3o + 2L + m\gamma.$$

Как и в алгоритме для коммутативной операции, вычислительная нагрузка в этом алгоритме распределена по процессам неравномерно – операции локальной редукции выполняются только половиной процессов. Больше всего вычислительной нагрузки приходится на корневой процесс, причем в качестве такового реко-

мендуется выбирать процесс 0. Это обусловлено тем, что итоговый результат операции формируется в процессе 0 и по окончанию фазы обменов передается корневому процессу.

Полученные выражения (в частности, оценки числа операций send/recv) могут быть использованы и для оптимизации формирования расписаний неблокирующих коллективных операций MPI 3 (non-blocking collective schedule) в сетевых адаптерах, допускающих выгрузку коллективных операций (offload collectives).

4. Алгоритм бинарного дерева. Алгоритм бинарного дерева организует процессы в сбалансированное по числу узлов бинарное дерево. Каждый процесс, зная свой номер и число процессов, определяет свое положение в дереве (рис. 4).

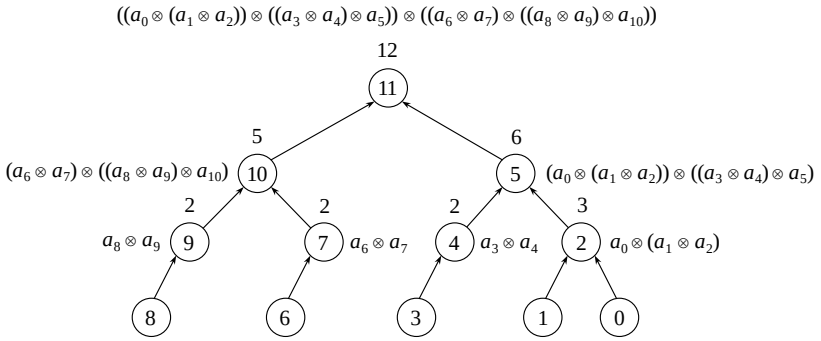

Рис. 4. Сбалансированное по числу узлов бинарное дерево обменов: $P = 12$, над узлами записано число процессов в поддереве, справа от узла – результат локальной редукции, a_i – сообщение процесса i

Дерево формируется так, чтобы число узлов в левом и правом поддеревьях отличалось не более чем на единицу. Корнем дерева является процесс с номером $P - 1$. В правое поддерево корня распределяется $\lfloor P/2 \rfloor$ узлов, в левое поддерево: $P - \lfloor P/2 \rfloor - 1$. Таким образом, при четном P число процессов в правом поддереве на единицу больше числа процессов в левом поддереве. Левым дочерним процессом корня становится узел $P - 2$, правым: $\lfloor P/2 \rfloor - 1$. Процедура распределения процессов продолжается рекурсивно для левого и правого поддеревьев. Если некоторому узлу назначен один дочерний элемент, то он становится его левым ребенком.

Случай 1. Корневым выбран процесс $P - 1$. Время выполнения алгоритма определяется временем корневого процесса

$$t_{root}(m) = \max\{o + L, m\lambda\} + (\lfloor \log_2 P \rfloor)(\max\{o + m\gamma, g\} + o + m\gamma) + (\lfloor \log 2\, P \rfloor - 1)(o + L).$$

Случай 2. Корневым выбран процесс $P \neq 1$. Если корневым является процесс, отличный от $P - 1$, то время выполнения алгоритма увеличится на время $2o + L$ передачи результата от процесса $P - 1$ корневому $root$:

$$t_{root}(m) = t_{P-1} + 2o + L.$$

В данном алгоритме перед началом фазы обменов процессы выполняют порядка $O(\log P)$ операций для поиска своей позиции в дереве. Алгоритм характеризуется хорошей масштабируемостью – время выполнения имеет порядок роста $O(\log P)$. Процессы загружены вычислениями неравномерно – в редукции участвует $P/2$ процессов, $P/4$, …, 1. В качестве корневого рекомендуется выбирать процесс с номером $P - 1$.

5. Алгоритм параллельных цепочек.

5.1. Алгоритм с фиксированным числом цепочек. Алгоритм k параллельных цепочек (k-chain algorithm) организует процессы в k цепочек (конвейеров), которые передают свои результаты корневому процессу с номером $root$, где k – это фиксированный параметр алгоритма. Например, в библиотеке Open MPI по умолчанию значение $k = 4$ (подсистема коллективных операций tuned) [18]. Если число процессов $P - 1$ не кратно значению k, то остаток $(P - 1)\ \%\ k$ процессов распределяется по первым $(P - 1)\ \%\ k$ цепочкам, в каждую добавляется по одному процессу. На рис. 5 первые две цепочки получили по одному дополнительному процессу.

При $k = 1$ алгоритм параллельных цепочек сводится к *конвейерному* алгоритму (pipeline reduce), а при $k = P - 1$, к алгоритму *плоского дерева* (flat/liner tree).

Будем называть цепочку *короткой*, если она содержит $\lfloor (P - 1)/k \rfloor$ процессов. Аналогично, *длинной* будем называть цепочку, если она содержит $\lfloor (P - 1)/k \rfloor + 1$ процессов. Число длинных цепочек равно $(P - 1)\ \%\ k$, количество коротких есть $k - (P - 1)\ \%\ k$.

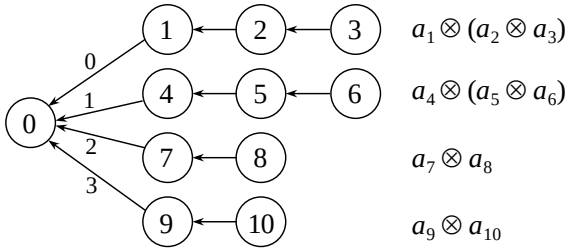

Рис. 5. Дерево обменов алгоритма параллельных цепочек: $P = 11$, $k = 4$, $root = 0$ (стрелками показаны направления передачи сообщений, справа результаты редукций в цепочках)

Корневой процесс выполняет k приемов результатов цепочек и столько же локальных редукций. Сперва выполняется прием результатов от длинных цепочек, а затем от коротких. Некорневые процессы по своему номеру определяют свое положение в цепочке – номера следующего и предшествующего процессов.

Очевидно, что время выполнения корневого процесса определяет время работы всего алгоритма. На рис. 6 приведена диаграмма выполнения в модели LogP алгоритма для $P = 11$, $k = 4$. В момент времени $\lfloor (P - 1)/k \rfloor (2o + L + m\gamma)$ головные процессы длинных цепочек завершают выполнение локальной редукции и готовы передавать результат корневому процессу.

На рис. 6 головной процесс 1 завершил выполнение локальной редукции в момент времени $4o + 2L + 2m\gamma$.

Корневой процесс получает результат первой длинной цепочки и завершает выполнение локальной редукции в момент времени $\lfloor (P - 1)/k \rfloor (2o + L + m\gamma) + 2o + L + m\gamma$. Корневой процесс не может начать очередной прием ранее чем, через $\max\{o + m\gamma, g\}$ единиц времени.

Поэтому суммарное время $t(m)$ выполнения корневого процесса равно

$$t(m) = \lfloor (P - 1)/k \rfloor (2o + L + m\gamma) + (k - 2)\max\{o + m\gamma, g\} + 3o + L + 2m\gamma.$$

Из полученного выражения $t(m)$ видно, что алгоритм параллельных цепочек с фиксированным значением k характеризуется линейной зависимостью времени выполнения от числа P процессов. Другими словами, скорость роста функции $t(m)$ имеет порядок $\Omega(P)$, что ограничивает применимость данного алгоритма в большемасштабных ВС. Кроме этого, в корневом процессе целесообразно сперва принимать результаты коротких цепочек, а затем длинных.

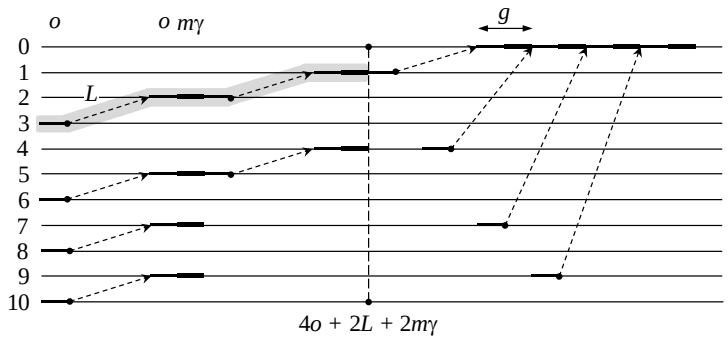

Рис. 6. Пространственно-временная диаграмма выполнения в модели LogP алгоритма параллельных цепочек (P = 11, k = 4, root = 0)

5.2. Алгоритм с оптимальным в модели LogP числом цепочек. Найдем число k цепочек при котором алгоритм характеризуется наименьшим в модели LogP временем выполнения. Запишем время алгоритма параллельных цепочек как функцию от параметра k

$$t(k) = ((P-1)/k + 1)(2o + L + m\gamma) + (k-2)\max\{o + m\gamma, g\} + o + m\gamma.$$

Далее будем исходить из предположения, что $o + m\gamma > g$. Это допущение не повлияет на окончательный результат. Для краткости изложения обозначим $a = 2o + L + m\gamma$, $b = o + m\gamma$. Тогда

$$t(k) = ((P-1)/k + 1)a + (k-1)b. \quad (1)$$

Найдем значение аргумента k, при котором функция $t(k)$ принимает минимальное значение.

$$t'(k) = -a(P-1)/k^2 + b, \quad -a(P-1)/k2 + b = 0, \quad k = \sqrt{a(P-1)/b}.$$

Последнее равенство справедливо, так как $a / b > 1$. Подставим найденное выражение для k в (1) и запишем время выполнения алгоритма с оптимальным числом цепочек как функцию от P

$$t(P) = \left(\frac{P-1}{\sqrt{a(P-1)/b}} + 1\right)a + \left(\sqrt{a(P-1)/b} - 1\right)b = 2\sqrt{ab}\sqrt{P-1} + a - b.$$

Если значения параметров o, L, γ заранее известны (предварительно измерены для заданной ВС), то оптимальное значение k определяется однозначно. На практике можно рекомендовать выбирать значение k^* числа цепочек исходя из следующего неравенства

$$k^* \geq \sqrt{P-1}.$$

На рис. 7 показана построенная в модели LogP зависимость времени $t(k)$ работы алгоритма от числа k цепочек. Нетрудно заметить, что кривые имею экстремумы в окрестностях точки $k^* = \sqrt{P-1}$.

На рис. 8 показана зависимость времени $t(k)$ выполнения алгоритма от числа k параллельных цепочек на вычислительном кластере с сетью связи InfiniBand QDR (48 процессов, 6 двухпроцессорных узлов, коммутатор Mellanox InfiniScale IV IS5030 QDR, сетевые адаптеры Mellanox MT26428 InfiniBand QDR, операционная система GNU/Linux CentOS 6.5 x86-64, библиотека MVAPICH2 2.2a).

Для каждого значения k рассчитывалось среднее время выполнения алгоритма по результатам 10 запусков (методика измерений описана в работах [19, 20]). Видно, что рекомендуемое значение $k^* = 7$ обеспечивает время выполнения алгоритма близкое к оптимальному (реализация алгоритма https://github.com/mkurnosov/reduce-kchain.git).

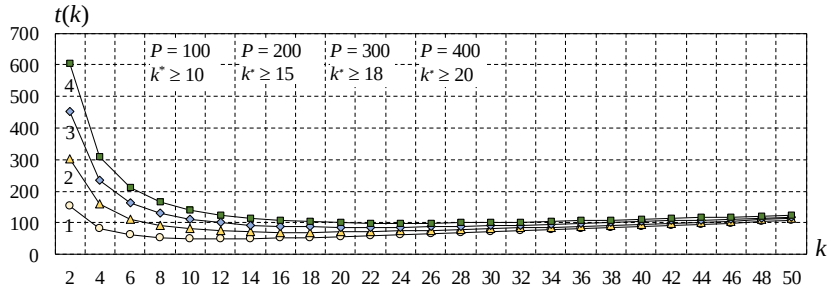

Рис. 7. Зависимость времени $t(k)$ выполнения алгоритма от числа k цепочек (модель LogP, $o + m\gamma > g$, $a = 3$, $b = 2$): 1) $P = 100$, $k^* = 10$; 2) $P = 200$, $k^* = 15$; 3) $P = 300$, $k^* = 18$; 4) $P = 400$, $k^* = 20$

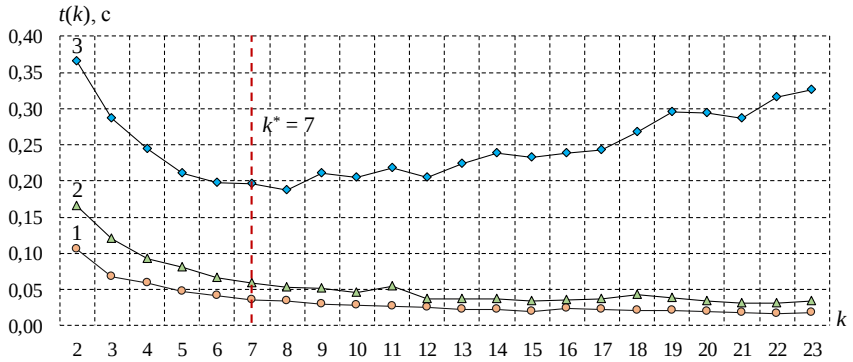

Рис. 8. Зависимость времени $t(k)$ выполнения алгоритма от числа k параллельных цепочек ($P = 48$, 6 двухпроцессорных узлов, сеть связи InfiniBand QDR, MVAPICH2 2.2a, суммирование элементов типа double): 1) $m = 1$; 2) $m = 1024$; 3) $m = 1\,048\,576$

Заключение. Для коллективной операции корневой редукции (all-to-one reduce) в модели параллельных вычислений LogP построены аналитические выражения (оценки) времени выполнения алгоритмов ее реализации. В отличие от известных работ, выражения построены для общих и частных (особых) случаев значений параметров вычислительной системы и коллективных операций. На примере алгоритма -параллельных цепочек продемонстрирован подход к построению оптимальных в модели LogP алгоритмов коллективных операций. Выбор конкретной математической модели параллельных вычислений обусловлен спецификой алгоритма и целевой ВС. Например, если алгоритм реализует группировку сообщений в пакеты больших размеров, то целесообразно использовать модель LogGP, которая в явном виде учитывает издержки на передачу сообщений больших размеров.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Хорошевский В.Г.* Распределенные вычислительные системы с программируемой структурой // Вестник СибГУТИ. – 2010. – № 2. – С. 3-41.
2. *Степаненко С.А.* Мультипроцессорные среды суперЭВМ. Масштабирование эффективности. – М.: Физматлит, 2016. – 312 с.
3. *Hoefler T., Moor D.* Energy, Memory, and Runtime Tradeoffs for Implementing Collective Communication Operations // Journal of Supercomputing Frontiers and Innovations. – 2014. – Vol. 1, No. 2. – P. 58-75.

4. *Balaji P., Buntinas D., Goodell D., Gropp W., Hoefler T., Kumar S., Lusk E., Thakur R., Traff J.* MPI on Millions of Cores // Parallel Processing Letters. – 2011. – Vol. 21, Issue 1. – P. 45-60.
5. *Alverson R., Roweth D., Kaplan L.* The Gemini System Interconnect // International Symposium on High Performance Interconnects. – 2010. – P. 83-87.
6. *Eisley N., Heidelberger P., Senger R.* The IBM Blue Gene/Q interconnection network and message unit // International Conference for High Performance Computing, Networking, Storage and Analysis. – 2011. – P. 1-10.
7. *Абрамов С.М., Заднепровский В.Ф., Шмелёв А.Б., Московский А.А.* СуперЭВМ ряда 4 семейства. СКИФ: штурм вершины суперкомпьютерных технологий // Вестник ННГУ. – 2009. – № 5. – P. 200-210.
8. *Левин В.К., Четверушкин Б.Н., Елизаров Г.С., Горбунов В.С., Лацис А.О., Корнеев В.В., Соколов А.А., Андрюшин Д.В., Климов Ю.А.* Коммуникационная сетьMBC-Экспресс // Информационные технологии и вычислительные системы. – 2014. – № 1. – С. 10-24.
9. *Симонов А.С., Макагон Д.В., Жабин И.А., Щербак А.Н., Сыромятников Е.Л., Поляков Д.А.* Первое поколение высокоскоростной коммуникационной сети Ангара // Наукоемкие технологии. – 2014. – Т. 15, № 1. – С. 21-28.
10. *Thakur R., Rabenseifner R., Gropp W.* Optimization of collective communication operations in MPICH // Int. Journal of High Performance Computing Applications. – 2005. – Vol. 19 (1). – P. 49-66.
11. *Bruck J. [et al.].* Efficient Algorithms for All-to-All Communications in Multiport Message Passing Systems // IEEE Trans. Parallel Distrib. Syst. – 1997. – Vol. 8 (11). – P. 1143-1156.
12. *Курносов М.Г.* Алгоритмы трансляционно-циклических информационных обменов в иерархических распределенных вычислительных системах // Вестник компьютерных и информационных технологий. – 2011. – № 5. – С. 27-34.
13. *Аветисян А.И., Гайсарян С.С., Иванников В.П., Падарян В.А.* Прогнозирование производительности MPI-программ на основе моделей // Автоматика и телемеханика. – 2007. – Вып. 5. – С. 8-17.
14. *Pjesivac-Grbovic J., Angskun T. [et al.].* Performance analysis of MPI collective operations // Cluster Computing. – 2007. – Vol. 10. – P. 127-143.
15. *Hoefler T., Schneider T., Lumsdaine A.* LogGOPSim – Simulating LargeScale Applications in the LogGOPS Model // Proc. of Int. Symposium on High Performance Distributed Computing. – 2010. – P. 597-604.
16. *Culler D., Karp R., Patterson D. [et al.].* LogP: Towards a Realistic Model of Parallel Computation // ACM SIGPLAN Notices. – 1993. – Vol. 28, No. 7. – P. 1-12.
17. *Kielmann T. [et. al.].* Fast Measurement of LogP Parameters for Message Passing Platforms // Proceedings of the 15 IPDPS 2000 Workshops on Parallel and Distributed Processing. – Springer Verlag, 2000. – P. 1176-1183.
18. *Fagg G., Pjesivac-Grbovic J., Bosilca G., Dongarra J., Jeannot E.* Flexible collective communication tuning architecture applied to Open MPI // Proc. of Euro PVM/MPI. – 2006. – P. 1-10.
19. *Worsch T., Reussner R., Werner A.* On Benchmarking Collective MPI Operations // Proceedings of the 9th EuroPVM/MPI Users' Group Meeting. – 2002. – P. 271-279.
20. *Курносов М.Г.* MPIPerf: пакет оценки эффективности коммуникационных функций библиотек стандарта MPI // Вестник Нижегородского университета им. Н.И. Лобачевского. – 2012. – № 5 (2). – С. 385-391.

REFERENCES

1. *Khoroshevskiy V.G.* Raspredelennye vychislitel'nye sistemy s programmiruemoy strukturoy [A distributed computing system with programmable structure], *Vestnik SibGUTI* [Vestnik SibGUTI], 2010, No. 2, pp. 3-41.
2. *Stepanenko S.A.* Mul'tiprotsessornye sredy superEVM. Masshtabirovanie effektivnosti [Multiprocessor supercomputer environment. Scaling efficiency]. Moscow: Fizmatlit, 2016, 312 p.
3. *Hoefler T., Moor D.* Energy, Memory, and Runtime Tradeoffs for Implementing Collective Communication Operations, *Journal of Supercomputing Frontiers and Innovations*, 2014, Vol. 1, No. 2, pp. 58-75.
4. *Balaji P., Buntinas D., Goodell D., Gropp W., Hoefler T., Kumar S., Lusk E., Thakur R., Traff J.* MPI on Millions of Cores, *Parallel Processing Letters*, 2011, Vol. 21, Issue 1, pp. 45-60.

5. *Alverson R., Roweth D., Kaplan L.* The Gemini System Interconnect, *International Symposium on High Performance Interconnects*, 2010, pp. 83-87.
6. *Eisley N., Heidelberger P., Senger R.* The IBM Blue Gene/Q interconnection network and message unit, *International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011, pp. 1-10.
7. *Abramov S.M., Zadneprovskiy V.F., Shmelev A.B., Moskovskiy A.A.* SuperEVM ryada 4 semeystva. SKIF: shturm vershiny superkomp'yuternykh tekhnologiy [Supercomputers series 4 family. SKIF: the summit supercomputer technologies], *Vestnik NNGU* **[**Vestnik of Lobachevsky University of Nizhni Novgorod], 2009, No. 5, pp. 200-210.
8. *Levin V.K., Chetverushkin B.N., Elizarov G.S., Gorbunov V.S., Latsis A.O., Korneev V.V., Sokolov A.A., Andryushin D.V., Klimov Yu.A.* Kommunikatsionnaya set'MVS-Ekspress [Communication, settings Express], *Informatsionnye tekhnologii i vychislitel'nye sistemy* [Information technologies and computing systems], 2014, No. 1, pp. 10-24.
9. *Simonov A.S., Makagon D.V., Zhabin I.A., Shcherbak A.N., Syromyatnikov E.L., Polyakov D.A.* Pervoe pokolenie vysokoskorostnoy kommunikatsionnoy seti Angara [The first generation of high-speed communications network of the Hangar], *Naukoemkie tekhnologii* [High-end technology], 2014, Vol. 15, No. 1, pp. 21-28.
10. *Thakur R., Rabenseifner R., Gropp W.* Optimization of collective communication operations in MPICH, *Int. Journal of High Performance Computing Applications*, 2005, Vol. 19 (1), pp. 49-66.
11. *Bruck J. [et al.].* Efficient Algorithms for All-to-All Communications in Multiport Message Passing Systems, *IEEE Trans. Parallel Distrib. Syst*., 1997, Vol. 8 (11), pp. 1143-1156.
12. *Kurnosov M.G.* Algoritmy translyatsionno-tsiklicheskikh informatsionnykh obmenov v ierarkhicheskikh raspredelennykh vychislitel'nykh sistemakh [Algorithms broadcast-cyclic information exchange in a hierarchical distributed computing systems], *Vestnik komp'yuternykh i informatsionnykh tekhnologiy* [Herald of computer and information technology], 2011, No. 5, pp. 27-34.
13. *Avetisyan A.I., Gaysaryan S.S., Ivannikov V.P., Padaryan V.A.* Prognozirovanie proizvo-ditel'nosti MPI-programm na osnove modeley [Performance prediction of MPI programs based on models], *Avtomatika i telemekhanika* [Automation and remote control], 2007, Issue 5, pp. 8-17.
14. *Pjesivac-Grbovic J., Angskun T. [et al.].* Performance analysis of MPI collective operations, *Cluster Computing*, 2007, Vol. 10, pp. 127-143.
15. *Hoefler T., Schneider T., Lumsdaine A.* LogGOPSim – Simulating LargeScale Applications in the LogGOPS Model, *Proc. of Int. Symposium on High Performance Distributed Computing*, 2010, pp. 597-604.
16. *Culler D., Karp R., Patterson D. [et al.].* LogP: Towards a Realistic Model of Parallel Computation, *ACM SIGPLAN Notices*, 1993, Vol. 28, No. 7, pp. 1-12.
17. *Kielmann T. [et. al.].* Fast Measurement of LogP Parameters for Message Passing Platforms, *Proceedings of the 15 IPDPS 2000 Workshops on Parallel and Distributed Processing.* Springer Verlag, 2000. pp. 1176-1183.
18. *Fagg G., Pjesivac-Grbovic J., Bosilca G., Dongarra J., Jeannot E.* Flexible collective communication tuning architecture applied to Open MPI, *Proc. of Euro PVM/MPI*, 2006, pp. 1-10.
19. *Worsch T., Reussner R., Werner A.* On Benchmarking Collective MPI Operations, *Proceedings of the 9th EuroPVM/MPI Users' Group Meeting*, 2002, pp. 271-279.
20. *Kurnosov M.G.* MPIPerf: paket otsenki effektivnosti kommunikatsionnykh funktsiy bibliotek standarta MPI [MPIPerf: the evaluation of the effectiveness of communication functions of MPI standard libraries], *Vestnik Nizhegorodskogo universiteta im. N.I. Lobachevskogo* **[**Vestnik of Lobachevsky University of Nizhni Novgorod], 2012, No. 5 (2), pp. 385-391.

Статью рекомендовал к опубликованию д.т.н. С.Н. Мамойленко.

Курносов Михаил Георгиевич – Институт физики полупроводников им. А.В. Ржанова СО РАН; e-mail: mkurnosov@gmail.com; 630090, г. Новосибирск, проспект Академика Лаврентьева, 13; тел.: +79139277213; к.т.н.; доцент; научный сотрудник Лаборатории вычислительных систем.

Kurnosov Mikhail Georgievich – Rzhanov Institute of Semiconductor Physics Siberian Branch of Russian Academy of Sciences; e-mail: mkurnosov@gmail.com; 13, Academician Lavrentyev Avenue, Novosibirsk, 630090, Russia; phone: +79139277213; cand. of eng. sc.; associate professor; research scientist; computer systems Laboratory.