УДК 004.932.4

Л.П. Фельдман, д-р техн. наук, проф.,
И.А. Назарова, канд. техн. наук, доц.
Донецкий национальный технический университет, г. Красноармейск
feldman@pmi.dgtu.donetsk.ua, ianazard@gmail.com

Эффективность реализации параллельных вычислений для кластерных систем на базе интерфейса MPI

В статье приводится технология получения теоретических и экспериментальных оценок динамических характеристик параллельных алгоритмов и их программных реализаций с использованием библиотеки передачи сообщений. Отдельно рассмотрены методы уменьшения трудоемкости коллективной операции «все-всем» интерфейса MPI под управлением OS Windows для топологий «звезда» и «кроссбар».

Ключевые слова: параллельные вычисления, кластерные системы, распределенная память, библиотека MPI, динамические характеристики.

Введение

Широкое использование кластерных вычислительных систем является одним из основных и стратегических направлений развития современной компьютерной техники. Под кластером обычно понимается множество отдельных гомогенных, а зачастую и гетерогенных компьютеров, объединенных в сеть при помощи специальных программно-аппаратных средств. При всем разнообразии кластерных систем (от количества отдельных компьютеров до используемых сетевых технологий и операционных систем) неизменным остается тот факт, что это системы с одинаковым способом организации памяти, а именно – распределенным доступом к локальным данным. На сегодняшний день основным стандартом организации распределенных вычислений в практических приложениях является интерфейс передачи сообщений или библиотека *MPI*.

В работах авторов [1-5] описаны параллельные алгоритмы и их реализации для численного решения задачи Коши на основе явных и неявных одношаговых схем с встроенными способами оценки локальной апостериорной погрешности для управления шагом интегрирования. Построены и исследованы схемы отображения таких алгоритмов на вычислительные структуры с распределенной памятью и различными топологиями межпроцессорного объединения. Исследованы вопросы эффективности и масштабируемости разработанных параллельных приложений.

В данной статье обсуждаются вопросы технологии получения теоретических и экспериментальных оценок динамических характеристик параллельных алгоритмов и их программных реализаций с использованием интерфейса передачи сообщений и исследование способов уменьшения трудоемкости коллективных операций передачи данных.

1 Особенности оценки динамических характеристик выполнения параллельных вычислений для систем с распределенной памятью

Основными общепризнанными динамическими характеристиками параллельных алгоритмов и их программных приложений являются коэффициенты ускорения, S и эффективности, E, которые, в свою очередь, зависят от:

1) времени на реализацию последовательного алгоритма, T_1;

2) времени реализации параллельных вычислений, $T_{p,comp}$;

3) времени на реализацию передачи данных, $T_{p,comm}$.

Рассмотрим основные факторы, влияющие на скорость выполнения вычислений и обменов для параллельных систем с распределенной памятью. Двумя такими факторами являются:

– операции с плавающей точкой;
– операции межпроцессорного обмена.

Количество операций с плавающей точкой в секунду (*Floating Point Operations Per Second – FLOPS*) является характеристикой производительности процессора и может быть определено экспериментально на основе различных тестов. Время выполнения n операций с плавающей точкой будем обозначать [6]:

$$T_F(n) = \frac{n}{FLOPS}, \qquad (1)$$

где *FLOPS* – производительность исследуемого процессора.

Пропускная способность сети и латентность являются характеристиками быстродействия сети. Под пропускной способностью B сети понимают количество информации, передаваемой между узлами сети в единицу времени (байт в секунду). Латентностью t_s называется время, затрачиваемое программным обеспечением и устройствами сети на подготовку к передаче информации по данному каналу. Временем пересылки одного элемента данных t_w называется время, затрачиваемое на передачу сообщения размером 4 байта в пакете с другими данными.

Производительность сети может быть определена экспериментально при помощи тестов вычислительной среды. Данные характеристики позволяет определить сложность одной операции пересылки данных как:

$$T_{send}(L) = t_s + L / B, \quad (2),$$

где L – количество передаваемых элементов данных. Однако удобнее использовать формулу, в которую явно входит время пересылки одного элемента данных t_w:

$$T_{send}(L) = t_s + L t_w. \quad (3)$$

Экспериментально измерим точность синхронизации *MPI*-программ. В данной работе для проведения вычислительных экспериментов использован вычислительный кластер с *Argonne National Library MPICH-2* – реализацией стандарта *MPI* для *OS* Windows [7]. Все измерения времени выполнения тех или иных алгоритмов измеряются с использованием функции *MPI_Barrier*, поэтому необходимо оценить точность реальной синхронизации алгоритмов с использованием данного вызова на данном кластере. Для проведения эксперимента выбран альтернативный метод синхронизации – по внешнему аппаратному прерыванию, который дает максимальную точность синхронизации, заведомо бо́льшую, чем *MPI* барьер. В качестве внешнего прерывания использовано прерывание *COM* порта и стандартный обработчик Windows. Системный вызов Windows *WaitCommEvent* задерживает работу программы до появления системного события. Программа продолжает выполнение, как только на *COM* порту будет получен байт данных. Точность синхронизации при помощи *MPI_Barrier*, по результатам внутреннего трассировщика *MPICH*, измеренная в *jumpshot* составляет 0.0002с., что видно из рис. 1. Точность синхронизации по внешнему прерыванию по результатам внутреннего трассировщика *MPICH* измеренная в *jumpshot* составляет 0.00006с., что показано на рис. 2.

Соотношение точности синхронизации по барьеру с точностью синхронизации по внешнему прерыванию показывает, что *MPI* барьер синхронизирует лишь в 3 раза менее точно, что является приемлемым результатом. Вопрос точности работы трассировщика *MPI*, а также точности синхронизации *MPI* программ требует более тщательного исследования. Проведенный эксперимент показывает, что в рамках данной работы на данном конкретном кластере результаты измерений задержек будут корректны с точностью до 5 знака после запятой.

Вычислительная сеть исследуемого кластера представляет собой коммутируемую Fast Ethernet сеть с топологией «звезда». В качестве центрального узла применен блокирующий коммутатор *Fast Ethernet Switch.*

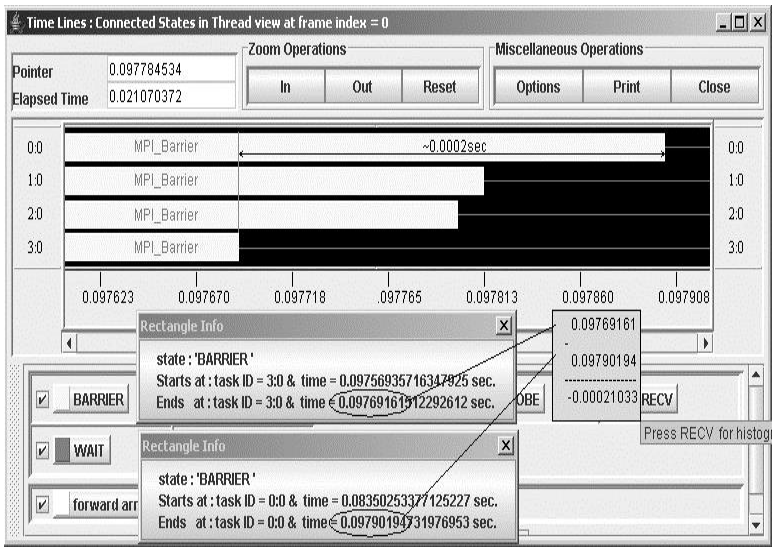

Рисунок 1 - Измерение точности синхронизации по *MPI_Barrier*

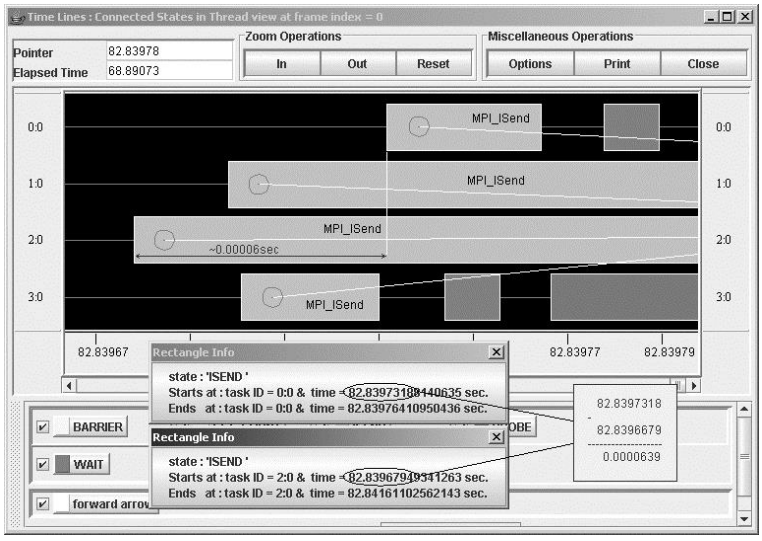

Рисунок 2 - Измерение точности синхронизации по прерыванию

Исследуем основные характеристики производительности данной сети. Прежде всего, следует определить пропускную способность коммутатора и показать, что он является блокирующим. Для этого следует сопоставить время выполнения единичной пересылки данных и массовой пересылки «каждый с каждым». Воспользуемся формулой для определения пропускной способности коммутатора для единичной пересылки:

$$B = 2KN / T \qquad (4)$$

и для массовой пересылки «каждый с каждым»

$$B = 2(p-1)KN / T\,, \qquad (5)$$

где T – время выполнения K обменов.

Измерив время T, получаем пропускную способность коммутатора. Как видим, пропускная способность коммутатора сохраняется при попытке выполнения параллельных обменов, что значит, что данный коммутатор является блокирующим и может выполнять коммутацию только двух узлов в данный момент времени. Найдем латентность и время пересылки элемента данных.

Для нахождения данных параметров воспользуемся формулой (3) и составим систему уравнений с двумя неизвестными t_w, t_s:

$$\begin{cases} t_s + t_w = 66.2196 \times 10^{-6} \\ t_s + 2048\,t_w = \dfrac{0.0935114307192}{2(100-4)} \cdot \end{cases}$$

Полученные характеристики вычислительной сети кластера могут быть использованы для анализа различных параллельных алгоритмов. Сопоставляя полученные значения с характеристиками, для *Gigabit Ethernet*, можно сделать вывод об адекватности полученных оценок. Латентность исследованной сети *Fast Ethernet* на 12мксек. больше чем для *Gigabit Ethernet*. Пропускная способность *Fast Ethernet* на порядок меньше, чем *Gigabit Ethernet.*

2 Анализ трудоемкости MPI обмена «каждый с каждым» и его оптимизация

Коллективная операция библиотеки MPI «каждый с каждым» является наиболее трудоемкой операцией обмена данными в кластерной системе. Вместе с тем она широко применяется во многих численных параллельных алгоритмах, таких, как матричные вычисления, решения систем уравнений: алгебраических, дифференциальных. В то же время трудоемкость данной операции является наиболее перспективной с точки зрения возможностей оптимизации, так как сильно зависит от топологии коммуникационной сети и ее характеристик.

Рассмотрим механизм работы MPI обмена «каждый с каждым». Вызов *MPI_Alltoall* выполняет последовательную пересылку локальных данных *i*-го процессора остальным $(p-1)$ процессорам. Время такого обмена составляет:

$$T_{Alltoall}(p,n) = (t_s + nt_w)(p-1)^2\,.$$

Под локальными данными понимается блок данных, расположенный на одном из процессоров. Под глобальными данными понимается совокупность локальных блоков данных всех процессоров. Требования массовой пересылки больших объемов данных могут превысить пропускную способность сети и появившиеся очереди снизят быстродействие всей системы в целом. Время латентности снижает

быстродействие пересылки и для малых объемов пересылаемых данных, быстродействие обмена «каждый с каждым» также окажется низким. Итак, основными замедляющими факторами являются:

– низкая пропускная способность канала при больших объемах пересылаемых данных;

– большое время латентности при малых объемах пересылаемых данных.

Исследуем возможности оптимизации обмена «каждый с каждым» в двух перечисленных случаях. Из (2) очевидно, что относительные задержки на латентность снизятся при увеличении объема пересылаемых данных. Попытка снизить количество обменов приводит к алгоритму обмена не локальными данными процессоров, а блоками глобальных данных, доступных разным процессорам.

Рассмотрим блочный алгоритм глобального распределения данных (обмен «каждый с каждым») *BAGDS*. Предлагаемый алгоритм обмена было бы некорректно называть «каждый с каждым», так как обмена каждого процессора с каждым фактически не происходит. Вместо этого обмениваются независимые пары процессоров блоками глобальных данных, отсутствующих в их локальной памяти.

Пусть имеется вычислительная система с количеством узлов *p*. Каждый узел содержит N/p элементов данных. Для определения текущего распределения данных каждый процессор имеет «маску распределения данных» *data_mask* и «маску пересылок» *send_mask*. Маска распределения данных представляет собой битовую таблицу. Элемент $data\_mask_{ij}=1$ означает, что процессор *i* имеет данные процессора *j* в локальной памяти. По маске распределения данных легко выяснить, какие данные необходимо получить определенному процессору для завершения сбора глобальных данных. На рис. 3 представлен пример структуры маски распределения данных для 5-ти процессоров.

	0	1	2	3	4
0	1				
1		1			
2			1		
3				1	
4					1

Рисунок 3 - Маска распределения данных для 5-ти процессоров

Элемент маски пересылок (рис. 4) $send\_mask_i=5$ показывает, что процессор *i* еще не готов передавать данные, т.к. процессора с рангом 5 не существует. Элементы маски $send\_mask_2=3$ и $send\_mask_3=2$ показывают, что процессоры с рангами 2 и 3 настроены на обмен данными.

0	1	2	3	4
5	5	3	2	5

Рисунок 4 - Маска пересылок

Реализация блочного алгоритма глобального распределения данных *BAGDS* заключается в выполнении следующих шагов:

1. Инициализация *data_mask* единичной матрицей I_N.

2. Если $data\_mask_{ij}=1$ для любого *i,j*, то выход. Иначе инициализация *send_mask* числом процессоров в системе.

3. По *send_mask* выбирается процессор *i*, не подготовленный к обменам.

4. По *data_mask* определяются данные, отсутствующие в его локальной памяти.

5. По *data_mask* находится процессор *j*, у которого есть данные отсутствующие в *i*, причем *j* также не подготовлен к обмену (см. 3)

6. Заносим в *data_mask* в строки *i* и *j* их объединение, заносим в $send\_mask_i=j$ и $siend\_mask_j=i$, что значит, что после обмена этих процессоров они будут располагать общими данными.

7. Если в *send_mask* остался один или ни одного неготового процессора, то выполнить обмен заданных пар процессоров и переход на 2. Иначе – новый этап, переход на 3.

Получим теоретическое время выполнения обменов по данному алгоритму для топологии «звезда». Обмен *MPI* «каждый с каждым» требует $(p-1)^2$ пересылок. Обмен по *BAGDS* займет не более чем $2(p/2)^2$ пересылок. Объем пересылаемых данных на каждом этапе в худшем случае будет удваиваться, что можно выразить формулой $(N/p)2^{level-1}$, N-общий объем данных, а *level* – номер этапа пересылок, который в свою очередь может быть выражен через номер пересылки как $level=i/p/2$, так как на каждом этапе во взаимных обменах участвует $p/2$ процессоров. Просуммируем все пересылки с учетом объемов пересылаемых данных, получая время выполнения глобального распределения данных на топологии «звезда».

Экспериментально, оценим влияние латентности и числа узлов на время выполнения операции обмена в условиях данной топологии. Показано, что с ростом объема распределяемых данных зона эффективности уменьшается, и

очевидно, что для некоторого объема данных *BAGDS* станет не эффективным. Проверим, как полученные зависимости согласуются с экспериментальными данными. На рис. 5 экспериментальные зависимости времени обмена от размерности задачи для алгоритмов *BAGDS* и *Alltoall*. Теоретически обоснованный предел эффективности *BAGDS* в районе сотни элементов данных подтверждается экспериментально.

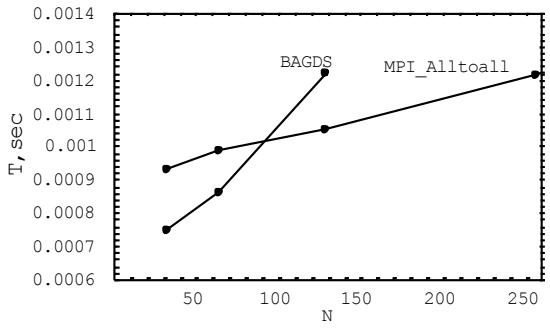

Рисунок 5 - Зависимость времени пересылки от объема распределяемых данных для топологии «звезда» с использованием коллективной операции *MPI* и алгоритма *BAGDS*

Рассмотрим теперь эффективность данного алгоритма на той топологии, для которой он был создан – «кроссбар» (см. рис. 6). Учитывая то, что независимые пересылки на данной топологии выполняются одновременно, время выполнения обменов (2) упростится. По полученным зависимостям можно сделать вывод, что алгоритм *BAGDS* эффективно работает на топологии «кроссбар» [8] и показывает ускорение в десятки раз по сравнению с обычным обменом *MPI_Alltoall*. Таким образом, было показано, что универсальный вызов интерфейса *MPI* для обмена по типу «каждый с каждым» не всегда дает максимальную эффективность, хотя для топологии «звезда» показывает лучшие результаты.

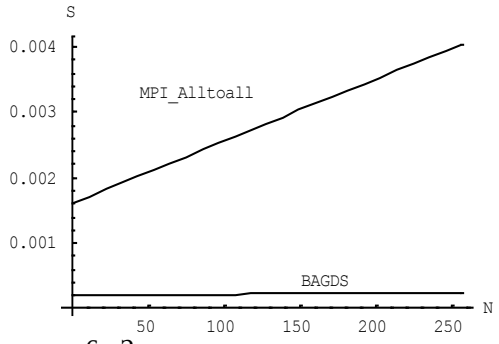

Рисунок 6 - Зависимость времени пересылки от объема распределяемых данных для топологии «кроссбар» с использованием коллективной операции *MPI* и алгоритма *BAGDS*

Предложенный алгоритм, позволяет значительно ускорить распределение глобальных данных за счет отказа от схемы обменов «каждый с каждым» и применения блочного подхода.

Заключение

Таким образом, в статье предложена и реализована методика измерения точности синхронизации параллельной программы с *MPI* под управлением *OS Windows*. Исследованы способы измерения латентности и пропускной способности вычислительного кластера с топологией вычислительной сети – «звезда», показана экспериментально малая пропускная способность такой топологии.

Предложен блочный алгоритм обмена данных «каждый с каждым», проведен теоретический и экспериментальный анализ эффективности предложенного алгоритма. Результаты анализа обобщены в виде рекомендаций по использованию данного алгоритма на сетях *Fast Ethernet* и *Gigabit Ethernet*, а также на топологиях «Звезда» и «Кроссбар».

Список использованной литературы

1. Фельдман Л.П. Паралельні однокрокові методи чисельного розв'язання задачі Коші: моногр. / Л.П. Фельдман, І.А. Назарова. – Донецьк: «ДВНЗ» ДонНТУ, 2011. – 185 с.: іл.

2. Фельдман Л.П. Современные параллельные методы численного решения задачи Коши: моногр. / Л.П. Фельдман, И.А. Назарова. – Донецк: ГВУЗ "ДонНТУ", 2013. – 206с.

3. Фельдман Л.П. Параллельные экспоненциальные методы решения задачи Коши с оценкой локальной апостериорной погрешности / Л.П. Фельдман, И.А. Назарова // Наукові праці Донецького національного технічного університету. Серія: Інформатика, кібернетика та обчислювальна техніка. – 2011. – Вып. 13 (185). – С. 7-12.

4. Фельдман Л.П. Параллельные алгоритмы экстраполяционных методов решения задачи Коши для компьютеров с распределенной памятью / Л.П. Фельдман, И.А. Назарова // Наукові праці Донецького національного технічного університету. Серія: Інформатика, кібернетика та обчислювальна техніка. – 2010. – Вып. 11(164). – С. 7-13.

5. Назарова И.А. Повышение эффективности решения жестких динамических задач для мультикомпьютеров с распределенной памятью / И.А. Назарова // Сборник трудов международной

конференции "Моделирование – 2008". – К.: Институт проблем моделирования в энергетике им. Г.Е. Пухова НАН Украины, 2008. – С. 471-476.

6. Андреев А.Н. Методика измерения основных характеристик программно-аппаратной среды / А.Н. Андреев, Вл. В. Воеводин. – М.: НИВЦ МГУ, 2003. – 10с: ил.

7. Корнеев В.Д. Параллельное программирование в MPI / В.Д. Корнеев. – Москва-Ижевск: Институт компьютерных исследований, 2003. – 304с.

8. Орлов С. Организация ЭВМ и систем: учеб. для ВУЗов / С. Орлов, Б. Цилькер. – Издательство: Питер, 2004. – 672 с.

Надійшла до редакції 15.03.2016

Л.П. ФЕЛЬДМАН, І.А. НАЗАРОВА
Донецький національний технічний університет

ЕФЕКТИВНІСТЬ РЕАЛІЗАЦІЇ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ ДЛЯ КЛАСТЕРНИХ СИСТЕМ НА БАЗІ ІНТЕРФЕЙСУ MPI

У статті наводиться технологія отримання теоретичних та експериментальних оцінок динамічних характеристик паралельних алгоритмів та їх програмних реалізацій із використанням бібліотеки передачі повідомлень. Окремо розглянуто методи зменшення трудомісткості колективної операції *«усі-усім»* інтерфейсу *MPI* під керуванням *OS Windows* для топології *«зірка»* та *«кросбар».*

Ключові слова: паралельні обчислення, кластерні системи, розподілена пам'ять, бібліотека MPI, динамічні характеристики.

L.P. FELDMAN, I.A. NAZAROVA
Donetsk National Technical University

THE EFFECTIVENESS OF IMPLEMENTATION OF PARALLEL COMPUTING FOR CLUSTER SYSTEMS BASED ON MPI

The article presents the technology of obtaining of theoretical and experimental estimations of the dynamic characteristics of parallel algorithms and their software implementations using the library message passing. We considered the methods to reduce the complexity of collective operations "all-to-all" interface *MPI* under control *OS Windows* for topology "star" and "crossbar". In the previous works the authors describe parallel algorithms and their implementation for the numerical solution of the Cauchy problem on the basis of explicit and implicit one-step methods with a posteriori local error estimation for management integration step. We constructed and investigated schemes mapping of such algorithms on computing structures with distributed memory, and various topologies of interprocessor association. The problems of efficiency and scalability of parallel applications are developed. This paper proposes a method to measure the accuracy of synchronization of the parallel program with *MPI* under control *OS Windows*. The accuracy of synchronization *MPI*-programs was measured experimentally. In the computational experiments a computing cluster was used with *Argonne National Library MPICH-2* - for the implementation of the *MPI* standard *OS Windows*. All measuring execution time of various algorithms is measured using *MPI_Barrier* function. For the experiment we selected alternative synchronization method - external hardware interrupts, which gives the most accurate synchronization, obviously larger than *MPI_Barrier*. Value barrier synchronization accuracy with synchronization accuracy external interrupt indicates that *MPI* barrier synchronizes only 3 times less accurately, which is an acceptable result. The accuracy of the *MPI*- tracer, as well as the accuracy of synchronization *MPI*-programs requires a more thorough investigation. The collective operation of the MPI-library «all-to-all" is the most time-consuming operation, the exchange of data in a cluster system. However, it is widely used in many numerical parallel algorithms, such as matrix calculation, solving systems of equations. At the same time, the complexity of this operation is the most promising in terms of the optimization possibilities, as much depends on the topology of the communication network and its characteristics. Mass transfer of large amounts of data may exceed the network capacity and reduce the performance of the whole system. The main inhibiting factors are: low bandwidth for large volumes of data transferred; great time latency for small amounts of data being transferred. The paper provides a block algorithm of data exchange "all-to-all", a theoretical and experimental analysis of the effectiveness of the proposed algorithm. The results of the analysis are summarized in the form of recommendations on the use of this algorithm on networks *Fast Ethernet* and *Gigabit Ethernet*, as well as on the topology "Star" and "Crossbar".

Keywords: parallel computing, cluster systems, distributed memory, library MPI, dynamic performance.