N. A. Moldovyan
Saint Petersburg Institute for Informatics and Automation of Russian Academy of Sciences
D. S. Budchan
Saint Petersburg Electrotechnical University «LETI»

CRYPTOGRAPHIC PROTOCOLS BASED ON SOLVING OF CUBIC EQUATIONS

There are considered protocols for public and deniable encryption with generating ciphertext in form of the set of the coefficients of cubic congruence modulo an integer that is difficult for factoring. It is proposed a general approach to constructing analogous protocols using modulo computations over binary polynomials. There are discussed peculiarities of solving cubic equations in finite binary fields. A general approach of the construction of an algorithm for solving cubic equations in finite binary fields is shown, which greatly simplifies the procedure of solving equations. It is proposed a novel digital signature protocol with cubic verification equation of general type, which is based on simultaneous encryption of both secret and fictitious messages. It is proposed to calculate the stability estimates of the developed algorithm of the electronic digital signature and algorithms based on calculations over binary polynomials.

Cryptography, encryption, deniable encryption, digital signature, cubic equation, binary finite field

УДК 004.272

В. В. Жариков, А. А. Пазников
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина)

Адаптивный алгоритм барьерной синхронизации в стандарте MPI на основе модели параллельных вычислений LogP

Рассматривается задача разработки адаптивного алгоритма барьерной синхронизации ветвей параллельных MPI-программ в распределенных вычислительных системах. Предложен алгоритм барьерной синхронизации, обеспечивающий субоптимальный выбор схемы реализации барьерной синхронизации. В процессе выбора учитывается время выполнения информационных обменов в модели параллельных вычислений LogP. Построены аналитические оценки времени выполнения распространенных алгоритмов барьерной синхронизации в модели LogP. Предложенный адаптивный алгоритм реализован в стандарте MPI. Приводятся результаты натурных экспериментов на кластерных вычислительных системах. Результаты экспериментов позволяют проследить зависимость между параметрами модели LogP, числом процессов и выбором алгоритма реализации барьерной синхронизации. Разработанный адаптивный алгоритм позволит сократить среднее время выполнения барьерной синхронизации на 4 %, по сравнению с существующими распространенными алгоритмами.

Коллективные обмены, барьерная синхронизация, распределенные вычислительные системы, LogP, MPI, параллельное программирование

В настоящее время для решения сложных задач в области науки и производства применяются распределенные вычислительные системы (ВС). В архитектурном плане распределенная ВС представляется множеством элементарных машин (ЭМ), взаимодействующих между собой через коммуникационную среду [1]. ЭМ может быть представлена как процессорным ядром, так и многоядерным SMP/NUMA-узлом, укомплектованным специализированными ускорителями (например, графическими процессорами). Количество ЭМ в системе может достигать нескольких миллионов. Так, например, суперкомпьютер Sunway TaihuLight, возглавляющий рейтинг TOP500, включает в себя более 10 млн процессорных ядер. Для выполнения параллельных программ на таких

системах в большинстве случаев используется модель передачи сообщений, представленная стандартом MPI (Message Passing Interface).

В значительной части существующих параллельных MPI-программ используются коллективные информационные обмены (групповые, collective), в которых участвуют все ветви (процессы) параллельной программы. На коллективные обмены приходится значительная доля суммарного времени выполнения программ [2]. Время реализации коллективных операций существенным образом влияет на масштабируемость параллельных программ, в связи с чем задача оптимизации коллективных обменов является актуальной. Основное направление работ по оптимизации коллективных обменов – разработка масштабируемых алгоритмов реализации коллективных операций на основе дифференцированных (двусторонних, точка-точка, point-to-point) обменов.

Одной из наиболее распространенных коллективных операций является барьерная синхронизация (barrier synchronization).

Барьерная синхронизация (барьер, barrier synchronization) – это коллективная операция, которая реализует ожидание процессами выполнения условия, когда каждый из них достигнет определенной точки в программе. Барьер (рис. 1) включает в себя 2 основных этапа: шаг захвата (capture), по достижении которого процесс переходит в состояние ожидания. После того как все процессы захвачены, выполняется процедура освобождения (release), которая выводит процессы из ожидания. На рис. 1 указаны номера параллельных процессов, участвующих в операции барьерной синхронизации.

Обзор существующих работ. К наиболее распространенным базовым алгоритмам барьерной синхронизации относятся Central Counter, Dissemination, Combining-Tree, Tournament и алгоритм Брукса. Большая часть современных работ направлена на развитие данных алгоритмов.

Авторы статьи [3] предлагают адаптивные версии алгоритма Combining-Tree, которые позволяют уменьшить объем накладных расходов на каждом узле дерева, включая захват и освобождение. В [4] предлагается оптимизированная версия алгоритма Брукса для числа процессов кратного двум. Алгоритмы барьерной синхронизации, представленные в [5], ориентированы на применение в сетях InfiniBand. В [6] предлагается метод на основе модели LogP параллельных вычислений с целью минимизации потребления энергии и метод масштабирования частоты процессоров для алгоритмов Tournament и Central Counter. В [7] предлагаются методы верификации алгоритмов Central Counter, Combining-Tree, Dissemination и Tournament. В [8] проводится анализ эффективности реализации барьерной синхронизации для языка Java.

Недостатком детерминированного выбора является то, что конкретные алгоритмы могут не обеспечивать минимальное время реализации операций барьерной синхронизации. Для этого может применяться адаптивный подход. При выборе алгоритма необходимо учитывать размер сообщений, интенсивность взаимодействия между параллельными ветвями и архитектурные свойства ВС [9].

В библиотеке MPICH применяется Dissemination, в Open MPI реализованы адаптивные схемы, которые осуществляют выбор между Central Counter и Combining-Tree, в зависимости от числа процессов. При большом количестве процессов применяется Combining-Tree.

В большинстве существующих адаптивных схем выполнения коммуникационных обменов для оценки времени выполнения применяется модель Хокни; в данной работе используется оценка эффективности на основе параметров модели LogP [10], которая позволяет с высокой точностью оценить время выполнения операций дифференцированных обменов в распределенных ВС. Оптимизация осуществляется с целью минимизации времени выполнения барьерной синхронизации.

Описание модели. В качестве основы для моделирования и оценки различных алгоритмов в описываемом случае используется модель LogP [10]. Данная модель позволяет с высокой точно-

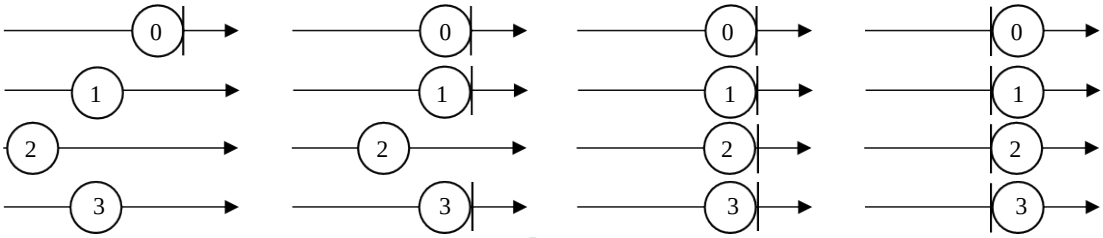

Рис. 1

стью оценить реальное время работы коллективных операций барьерной синхронизации. Основные параметры модели: L – латентность (latency) среды связи; o – накладные расходы времени (overhead) на передачу или прием сообщения; g – необходимый временной промежуток (gap) между двумя передачами или приемами сообщений; P – количество процессов в системе.

Используем обозначения: o_r – промежуток времени, когда процесс занят приемом сообщения; o_s – промежуток времени, когда процесс занят передачей сообщения [11]. Время для отправки и получения одного сообщения может быть аппроксимировано как $o_s + L + o_r$, а время отправки n сообщений может быть оценено как $o_s + (n-1)\max\{o_s, g\}$. Время для приема n сообщений (относительно первого отправленного пакета на стороне отправителя) можно смоделировать как $o_s + L + o_r + (n-1)\max\{o_s, g\}$.

Кроме того, введем и другие обозначения [12]:

$$f_r = \max\{o_r, g\},$$
$$f_s = \max\{o_s, g\},$$
$$t_r = \max\{f_r, o_s + L + o_r\} = \max\{\max\{o_r, g\}, o_s + L + o_r\} = \max\{g, o_s + L + o_s\},$$
$$t_s = \max\{g, o_s + L + o_r\},$$
$$f_r = f_s = o,$$
$$t_r = t_s = 2 \times o + L.$$

Адаптивный алгоритм барьерной синхронизации. Предложен адаптивный алгоритм субоптимального выбора алгоритмов барьерной синхронизации, учитывающий время выполнения информационных обменов при реализации барьерной синхронизации в модели параллельных вычислений LogP. Входными данными для алгоритма являются: коммуникатор MPI, параметры модели LogP (L, o_r, o_s, g, P) и массив BarrierAlgs. Данные параметры можно получить от системы мониторинга коммуникационной среды, например Netgauge.

Предложенный алгоритм включает в себя 3 основных шага:

1. Вычисление оценки времени реализации барьерной синхронизации каждого из рассматриваемых базовых алгоритмов.

2. Поиск минимального из значений времени и выбор алгоритма на основе результатов поиска.

3. Выполнение барьерной синхронизации на основе выбранного алгоритма.

Листинг 1. Псевдокод алгоритма AdaptiveBarrierLogP

```
Входные данные:
L, o, g, P – параметры модели LogP
BarrierAlgs – массив алгоритмов длины n.
    for i = 1 to n do
        t[i] = CompTime(L, o_r, o_s, P,
BarrierAlgs[i]) // сохранить в массиве время выполнения алгоритмов
    end for
    i* = argmin(t[i])  // получить индекс элемента массива с минимальным значением времени
        i = 1,…, n
    MPI_Barrier(BarrierAlgs[i*])  //  запустить выбранный алгоритм барьерной синхронизации
```

Аналитические модели алгоритмов. Оценим в модели LogP время реализации распространенных алгоритмов барьерной синхронизации.

Central Counter. Алгоритм Central Counter изображен на рис. 2. Левая часть обозначает фазу 1, в которой каждый некорневой процесс отправляет сообщение корневому процессу о том, что он достиг барьера. Если процессы отправляют сообщения одновременно, а их латентности равны между собой, то пакеты сериализуются в приемнике, поскольку принятое сообщение «блокирует» сетевой интерфейс на время f_r. На этом шаге корневой процесс является узким местом.

Фаза 2 начинается сразу после получения корневым процессом всех пакетов. Такая же блокировка происходит во время операции отправки, потому что каждый отправленный пакет «блокирует» отправителя на время f_s. Все остальные вычисления в модели LogP могут быть получены аналогично.

Фаза 1 завершается после

$$t' = o_s + L + (P-2) f_r + o_r.$$

Фаза 2 начинается с $T = t'$ и продолжается до времени выполнения последнего процесса. Время ее выполнения можно оценить следующим образом:

$$t'' = o_s + (P-2) f_s + L + o_r.$$

Процесс 0 завершил барьер после

$$t_0 = t' + o_s + (P-2) f_s = 2 o_s + o_r + L + (P-2) f_r + (P-2) f_s.$$

Процесс $i \in \{1, \ldots, P\}$ заканчивается после

$$t_i = t' + o_s + L + (i-1) f_s + o_r = 2 (o_s + L + o_r) + (P-2) f_r + (i-1) f_s.$$

Последний процесс (P) заканчивается, и для всего барьера получаем следующее время реализации:

$$r_t = 2 (o_s + L + o_r) + (P-2) f_r + (P-2) f_s.$$

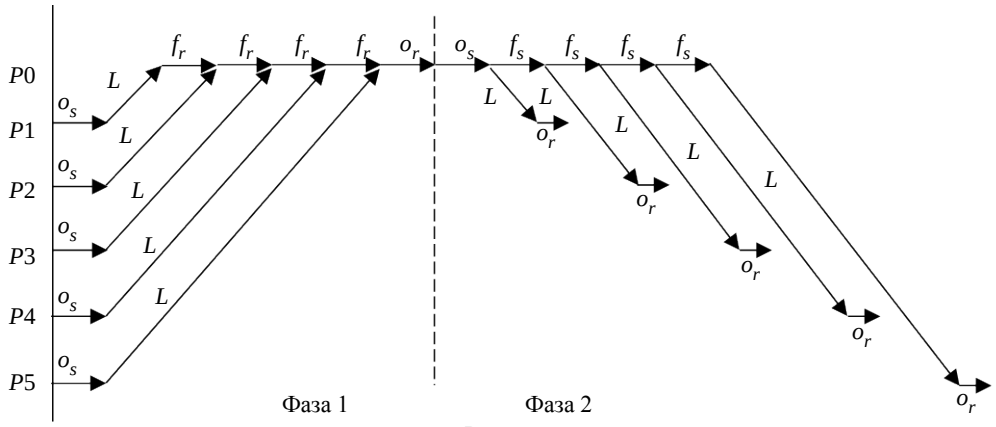

Рис. 2

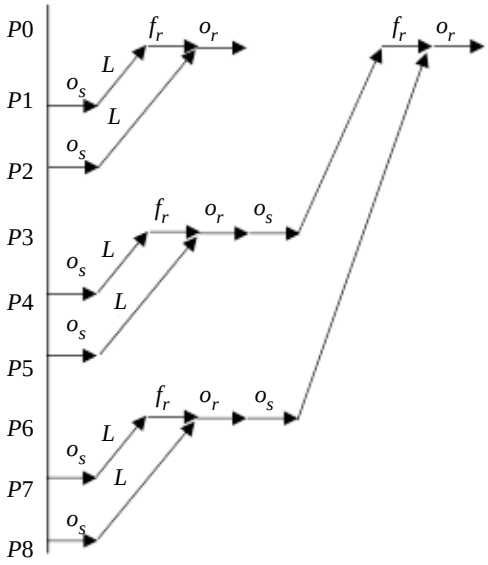

Рис. 3

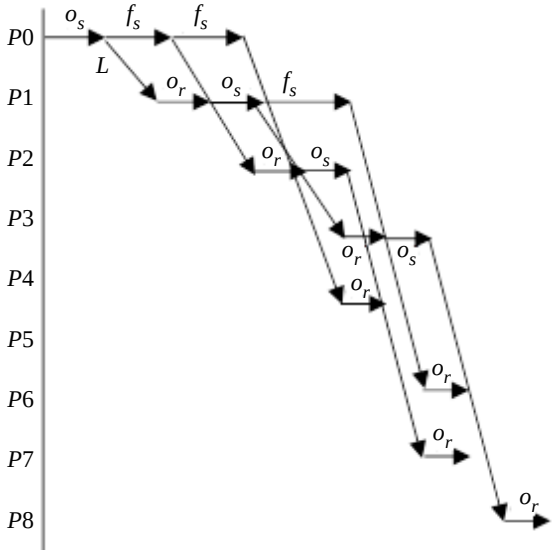

Рис. 4

Combining-Tree. Алгоритм Combinig-Tree имеет сложность $O(n\log_n P)$. Основная его идея состоит в том, чтобы разбить операцию барьерной синхронизации на дерево барьеров меньшего размера и объединить в себе ветви запросов и ветви уведомлений.

Каждому процессу присваивается уникальный узел дерева, который связан в дерево захвата по родительской ссылке и в дерево освобождения набором дочерних ссылок. Родитель уведомляет каждого из своих потомков, устанавливая флаг в узлах, соответствующих им. Потомок, в свою очередь, устанавливает флаг в родительском узле, сигнализируя о его попадании в барьер.

Combinig-Tree барьер характеризуется количеством процессов P, а также основанием n, характеризующим число потомков каждого узла (в случае двоичного дерева).

Согласно модели LogP время выполнения первой фазы (сбор данных на корневом процессе) получим по формуле $t' = (o_s + L + f_r \times (n-2) + o_r) \log_n P$ (рис. 3).

Время выполнения второй фазы алгоритма (рис. 4), использующего в своей основе бинарное дерево:

$$t'' = o_s + (\log_2 P - 1) \max \{g, o_s + L + o_r\} + L + o_r = o_s + (\log_2 P - 1)\, t_s + L + o_r.$$

Суммарное время реализации:

$$r_t = (o_s + L + f_r\,(n-2) + o_r) \log_n P + o_s + (\log_2 P - 1)\, t_s + L + o_r.$$

Dissemination Barrier. Алгоритм Dissemination имеет сложность $O(\log_2 P)$ и относится к классу барьеров, которые теоретически быстрее рассматриваемых в статье существующих алгоритмов обеспечивают реализацию барьерной синхронизации. Этот алгоритм получил свое название от того, что в процессе работы распространяет информацию среди набора процессов (рис. 5). Каждый процесс вращается вокруг выделенной ему переменной и сигнализируется другим процессом. Каждый процесс проходит через $\log N$ раундов, где N – количество процессов. В конце

раунда процесс знает, что другие процессы достигли барьера, и готов перейти к следующему раунду.

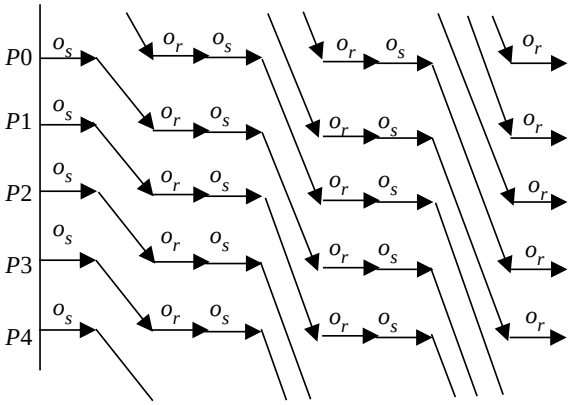

Рис. 5

Время выполнения алгоритма в модели LogP можно оценить следующим образом:

$$r_t = \max\{f_r, f_s, o_s + L + o_r\} \log_2 P = \\ = \max\{t_r, t_s\} \log_2 P.$$

Эксперименты. Эксперименты производились на кластерных ВС информационно-вычислительного центра Новосибирского национального исследовательского государственного университета. Использовался вычислительный кластер, укомплектованный 30 узлами с двумя 12-ядерными процессорами Intel Xeon E5-2680v3 с частотой 2500 МГц, ОЗУ 192 Гбайт. Операционная система: SUSE Linux Enterprise Server. Компилятор: GCC 4.8.5. Библиотека Open MPI 1.10.2.

Тестовая программа представляет собой циклический запуск функции барьерной синхронизации. Между вызовами барьерной синхронизации установлена задержка продолжительностью 1000 ± ± 100 мс. Количество итераций цикла – 400. Суммарное время выполнения барьерной синхронизации для текущего числа процессов определялось как среднее арифметическое результатов замера времени прохождения каждого из четырех барьеров.

В процессе проведения экспериментов количество процессов варьировалось от 2 до 100 с шагом 2. В качестве показателя эффективности алгоритма использовалось время t выполнения барьерной синхронизации. Алгоритм с наименьшим временем выполнения барьерной синхронизации является наиболее эффективным.

Результаты экспериментов можно увидеть на рис. 6, 7, где отражен один и тот же эксперимент с той лишь разницей, что на рис. 7 не показан график Central-Counter, чтобы повысить читаемость остальных. Central-Counter барьер заметно уступает алгоритмам Dissemination, Combinig-Tree и AdaptiveBarrierLogP при увеличении числа процессов. В свою очередь Dissemination и Combinig-Tree демонстрируют близкие результаты, но последний уступает по времени выполнения. Разработанный AdaptiveBarrierLogP алгоритм в большинстве экспериментов показывает результаты, лучшие в среднем на 4 %, что позволяет положительно судить о его эффективности. Чаще всего AdaptiveBarrierLogP алгоритм в процессе работы выбирал между реализациями Dissemination (96 %) и Combinig-Tree (3 %), частота выбора реализации барьерной синхронизации Central-Counter составила 1 % от общего числа. При числе процессов, меньшем 20 (рис. 8), превалирует выбор алгоритмом AdaptiveBarrierLogP реализации Combinig-Tree, в остальных случаях более эффективным оказывается выбор Dissemination. На рис. 9 представлено время выполнения операции, полученное на основе представленных в данной статье аналитических формул, для следующих значений параметров модели LogP (L = 125.6, o_r = 123.8, o_s = = 0.43, g = 0.22, $P \in \{2, ..., 100\}$).

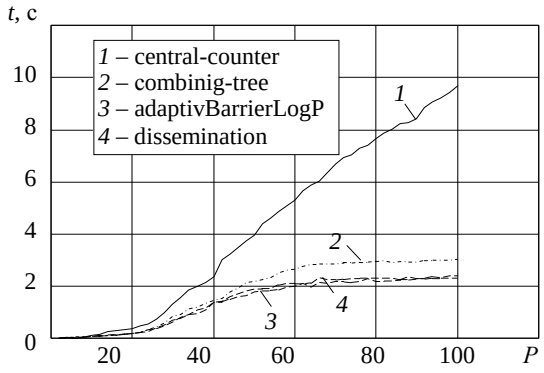

Рис. 6

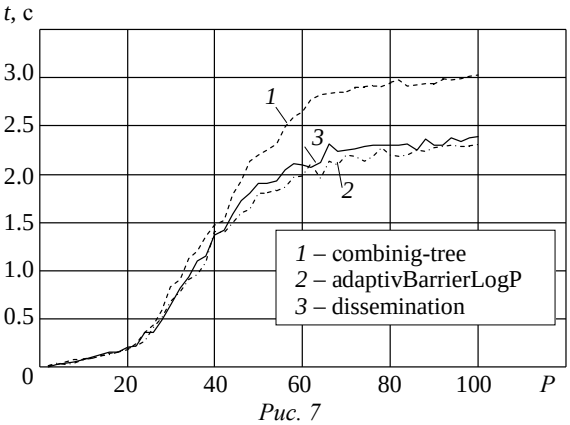

Рис. 7

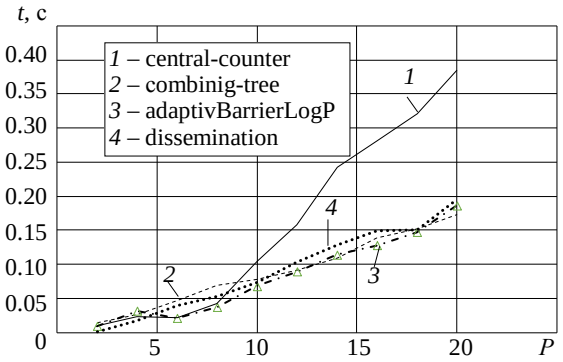

Рис. 8

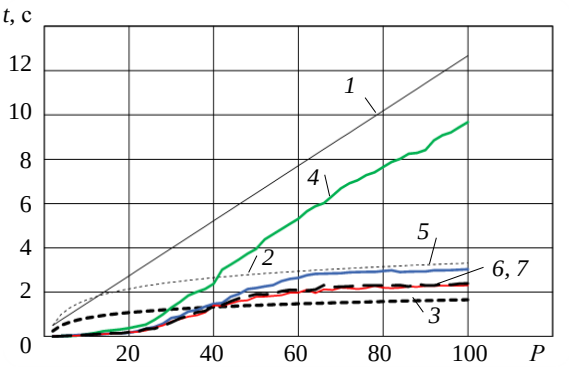

Рис. 9

На основе критерия χ-квадрат вычислены вероятности соответствия экспериментальных данных: p_1 = 09950419 (Combinig-Tree); p_2 = = 0.999992 (Dissemination) и p_3 = 0.296601 (Central-Counter). Данные вероятности близки к 1, что намного больше уровня значимости α = 0.05, при котором событие уже можно считать неслучайным. Поэтому можно утверждать, что экспериментальные данные не противоречат аналитическим и даже соответствуют нормальному закону распределения.

Разработан адаптивный алгоритм AdaptiveBarrierLogP субоптимального выбора алгоритмов барьерной синхронизации, учитывающий время выполнения информационных обменов в модели параллельных вычислений LogP. Построены аналитические оценки времени выполнения алгоритмов барьерной синхронизации в модели LogP. Приводятся результаты натурных экспериментов на базе действующих кластерных ВС.

Полученные на основе критерия χ-квадрат значения вероятностей близки к 1, что соответствует нормальному закону распределения и подтверждает корректность экспериментальных данных.

Результаты исследования созданного адаптивного алгоритма барьерной синхронизации на кластерной ВС показали, что среднее время реализации выполнения синхронизации с учетом параметров модели LogP уменьшилось в среднем на 4 % по сравнению с использованием неадаптивных реализаций. При числе процессов, меньшем 20 (рис. 8), превалирует выбор алгоритма Combinig-Tree, в остальных случаях более эффективным оказывается выбор Dissemination.

СПИСОК ЛИТЕРАТУРЫ

1. Хорошевский В. Г. Архитектура вычислительных систем. М.: Изд-во МГТУ им. Н. Э. Баумана, 2008. 520 с.

2. Paul S., Sandra W. Berkeley's Dwarfs on CUDA. RWTH Aachen University, 2011. 27 p.

3. Scott M., Mellor-Crummey J. Fast contention-free combining tree barriers for shared-memory multiprocessors // Intern. J. of Parallel Programming. 1994. Vol. 22, № 4. P. 449–481.

4. Hensgen D., Finkel R., Manber U. Two algorithms for barrier synchronization // Intern. J. of Parallel Programming. 1988. Vol. 17, № 1. P. 1–17.

5. Hoefler T. Fast Barrier Synchronization for InfiniBand // In Proc. of the 20th IEEE Intern. Parallel & Distributed Proc. Symp. (IPDPS), CAC'06 Workshop. Rhodes, Greece, 2006. P. 272– 275.

6. Juan C., Yong D. Energy optimization of representative barrier algorithms // J. of Central South University. 2012. Vol. 19, № 2. P. 2823–2831.

7. Malkis A., Banerjee A. Automation in the Verification of Software Barriers // Intern. J. of Parallel Programming. 2014. Vol. 52, № 3. P. 275–329.

8. Carwyn B., Mark B. Barrier Synchronisation in Java // Technical report, High-End Computing program (UKHEC). 2005. № 1. P. 1–25.

9. A Survey of Barrier Algorithms for Coarse Grained Supercomputers / T. Hoefler, S. Mehlan, A. Mietke, T. Rehm, S. Wolfgang // Technical report 3. University of Chemnitz (Germany). 2004. № 1. P. 1–20.

10. Performance analysis of MPI collective operations / J. Pješivac-Grbović, T. Angskun, G. Bosilca, E. Graham, G. Edgar, J. Jack // Cluster Computing. 2007. Vol. 10, № 2. P. 127–143.

11. Kielmann T., Bal H., Verstoep K. Fast measurement of LogP parameters for message passing platforms // Intern. Parallel and Distributed Proc. Symp. Springer, Berlin, Heidelberg, 2000. P. 1176-1183.

12. Hoefler T., Wolfgang R.. A practical Approach to the Rating of Barrier Algorithms using the LogP Model and Open MPI // Conf. Parallel Processin. Source: IEEE Xplore, 2005. P. 562–569.

V. V. Zharikov, A. A. Paznikov
Saint Petersburg Electrotechnical University «LETI»

ADAPTIVE ALGORITHM OF BARRIER SYNCHRONIZATION IN THE MPI STANDARD ON THE BASIS OF PARALLEL COMPUTING MODELS LOGP

The problem of developing an adaptive algorithm for barrier synchronization of branches of parallel MPI-programs in distributed computing systems is considered. An algorithm of barrier synchronization providing a suboptimal choice of the scheme for implementing barrier synchronization is proposed. During the selection process, the time of execution of information exchanges in the LogP parallel computing model is taken into account. Analytic estimates of the execution time of the barrier synchronization algorithms in the LogP model are constructed. The proposed adaptive algorithm is implemented in the MPI standard. The results of field experiments on cluster computer systems for analyzing the efficiency of the created algorithms are presented. The results of the experiments allow us to trace the relationship between the parameters of the LogP model, the number of processes and the choice of algorithms for the implementation of barrier synchronization. The developed algorithm made it possible to achieve a 4% decrease in the average implementation time for the implementation of barrier synchronization.

Collective exchanges, barrier synchronization, distributed computing systems, LogP, MPI, parallel programming

УДК 621.391

О. О. Луковенкова
Камчатский государственный университет им. Витуса Беринга

А. Б. Тристанов
Институт космофизических исследований и распространения радиоволн ДВО РАН

В. В. Геппенер
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина)

Моделирование частотно-временной структуры геоакустических импульсов с использованием методов интеллектуального анализа

Рассмотрено моделирование частотно-временной структуры заданного класса геоакустических импульсов. Для реализации процесса моделирования предложено использование комбинации методов частотно-временного, статистического и интеллектуального анализа, а также методов разреженной аппроксимации. Частотно-временное представление сигнала строится с использованием алгоритма адаптивного согласованного преследования. Показано разбиение построенных представлений на классы методами иерархической кластеризации. Получены и проанализированы 4 класса геоакустических импульсов. Предложено выявлять внутренние закономерности структуры заданного класса импульсов с использованием алгоритмов поиска ассоциативных правил и статистического анализа. Полученные алгоритмы затем используются для моделирования импульсов геоакустической эмиссии. Разработанные алгоритмы анализа и моделирования геоакустических импульсов реализованы на языке программирования MATLAB. Созданная система протестирована на типовых геоакустических импульсах характерной формы. Показано соответствие реальных и модельных импульсов, что доказывает корректность выявленных закономерностей.

Геоакустическая эмиссия, геоакустический импульс, частотно-временная структура, интеллектуальный анализ сигналов, разреженная аппроксимация

Под акустической эмиссией понимаются упругие колебания, возникающие в результате дислокационных изменений в твердых телах. Свойства регистрируемого при этом импульсного излучения напрямую зависят от характеристик порождающего пластического процесса [1]–[3]. В зависи-