УДК 621.396.664

Informatics and Mathematical Methods in Simulation
Vol. 4 (2014), No. 2, pp. 105-110

ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ НА КЛАСТЕРАХ

В.Д. Павленко

Одесский национальный политехнический университет,
пр. Шевченко, 1, Одесса, 65044, Украина; e-mail: pavlenko_vitalij@mail.ru

Рассматриваются принципы организации кластерных вычислений с помощью неявного распараллеливания, основанного на понятии заказа, а также особенности новой, альтернативной MPI, технологии распараллеливания вычислений, позволяющей для достаточно широкого класса вычислительных алгоритмов естественным образом переходить от существующих последовательных реализаций к параллельным.

Ключевые слова: параллельные вычисления, кластеры, технология программирования параллельных вычислений, архитектура программного обеспечения

Введение

Параллельные вычисления достаточно динамично развиваются в последнее время [1-4], что обусловлено, в первую очередь, появлением все более сложных задач, решение которых на современных последовательных ЭВМ за приемлемое время получить невозможно. К ним относятся задачи идентификации и моделирования нелинейных динамических систем на основе моделей Вольтерра [5-7], задачи информационной оптимизации, возникающие при проектировании систем модельной диагностики [8, 9], цифровой фильтрации изображений [10] и др. В данное время интерес к параллельным вычислениям также во многом связан со все большим распространением многоядерных процессоров [11].

Для эффективного создания параллельных приложений следует учитывать особенности применяющейся параллельной архитектуры [2]. Здесь в качестве такой архитектуры выбраны кластеры. Кластер представляет собой набор компьютеров, процессоры которых не имеют общей памяти, а обмениваются данными через сеть [12]. Главным преимуществом кластеров является их низкая стоимость, поскольку во многих случаях используются обычные массовые ПК и традиционные сетевые технологии; кластер может быть просто временно организован из простаивающих компьютеров какой-либо сети. Главным недостатком кластеров является скорость передачи данных между отдельными процессорами, которая является относительно низкой из-за отсутствия общей памяти. Это приводит либо к необходимости использования альтернативных сетевых технологий, либо к сужению класса эффективно реализуемых алгоритмов.

Одной из задач, полностью пока не решенных, является задача создания инструментальных средств параллельного программирования. Одним из основных препятствий для создания подобных инструментальных средств является сложность обнаружения участков кода в программе, которые могут выполняться параллельно. В случае же, если в используемой параллельной архитектуре процессоры не имеют общей памяти, реализация обмена данными обычно тоже является одной из задач прикладного программиста.

Существуют два основных подхода для снятия этого препятствия. Первый состоит во внесении в язык программирования средств декларирования потенциального параллелизма, примерами его применения являются технология OpenMP, языки программирования НОРМА и Haskell. Второй – в предоставлении пользователю мощных низкоуровневых средств организации взаимодействия процессоров, примерами его применения являются технология MPI и инструментарий Globus Toolkit. В данное время преимущественно используется второй подход, поскольку он позволяет добиться более высокой эффективности параллельных приложений за счет предоставления прикладному программисту более мощных средств контроля над используемой параллельной вычислительной системой. Обратной стороной данного подхода является более высокая сложность разработки и отладки параллельних приложений и перенос акцента в процессе разработки приложения с реализации алгоритма прикладной задачи к реализации работы со средствами, предлагаемыми выбранной технологией параллельного программирования.

Первый подход предпочтителен в задачах со сложным взаимодействием процессоров в ходе выполнения, возможностью внесения существенных изменений в логику распараллеливания в процессе развития программ. Преимущества первого подхода являются еще более заметными в случае отсутствия априорных сведений априорных сведений о составе кластера, поскольку наличие в программе именно сведений о потенциальном параллелизме, а не конкретных операций взаимодействия процессоров, позволяет реализовать достаточно сложные средства балансировки нагрузки.

Принципы технологии транспарентного распараллеливания на основе заказов

Предположим, что в распараллеливаемой программе выделен набор процедур, в процессе работы которых не модифицируются никакие переменные, кроме параметров и временно создаваемых структур данных, а входные и выходные параметры передаются по значению. Выполнение программы при этом должно сводиться к выполнению одной из этих процедур. Подобное предположение налагает достаточно существенные ограничения на программу. Например, оно запрещает всякую работу с глобальными переменными и устройствами ввода-вывода. Однако, как будет показано далее, подобное предположение относительно программы может быть ослаблено. Так, процедурам можно дать возможность работы с глобальными переменными или устройствами ввода-вывода при соблюдении некоторых условий.

Для иллюстрации предлагаемой технологии рассмотрим участок программы, схематическое изображение которого представлено на рис. 1, где прямоугольником обозначается время выполнения процедуры, отсчитываемое слева направо. Для обозначения вложенных процедур используются вложенные прямоугольники. Длины прямоугольников и их частей пропорциональны времени, затрачиваемому на выполнение соответствующих фрагментов программы. Линиями изображаются связи, которые объединяют моменты вычисления некоторых данных с моментами, когда они впервые используются в дальнейших вычислениях, при этом изображая только связи для данных, вычисляемых во вложенных процедурах, и используемых в основной процедуре или другой вложенной процедуре.

Первый принцип, на котором основан предлагаемый метод распараллеливания, заключается в понятии заказа на вычисления. Заказ на вычисления вводится как единица выполняемой на одном из компьютеров кластера работы, т.е. объем работы, который обязан быть выполнен на одном из компьютеров кластера полностью и не может быть разбит на более мелкие части. В качестве такого объема работы вводится выполнение одной процедуры без выполнения тех процедур, которые она вызывает: выполнение каждой из них выделяется в отдельный заказ. Для того чтобы определить

точку старта программы, одна из процедур должна быть описана как главная. Именно через параметры этой процедуры должны передаваться исходные данные и результаты вычислений.

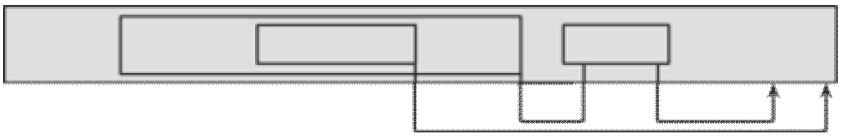

Рис.1. Схематическое изображение участка программы для распараллеливания

Этот принцип иллюстрирует рис. 2 (считается, что возникающие за время работы четыре заказа будут выполняться на четырех различных процессорах, а их выполнение будет начинаться сразу же). Для наглядности показаны связи по данным между частями процедур. Прерывистыми линиями выделены части вычислительного процесса, производимые на одном процессоре.

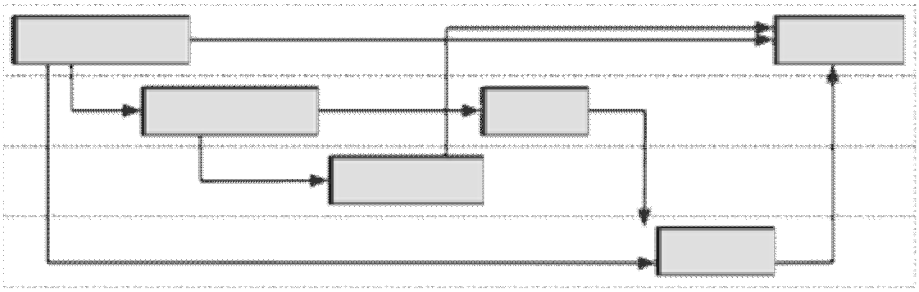

Рис. 2. Иллюстрация первого принципа технологии распараллеливания

В значительном числе алгоритмов существует отрезок времени между получением некоторых данных и первым их использованием в дальнейших вычислениях; часто он может быть увеличен путем внесения изменений в порядок вычислений, не изменяющих результат. Наличие таких промежутков, связано с возможностью распараллеливания алгоритма, поскольку при их отсутствии работа данного алгоритма не может быть распараллелена никакими средствами. Второй принцип заключается в следующем. При традиционном подходе при вызовах процедур вызывающая процедура продолжает свою работу только после окончания работы вызываемой процедуры – или, иными словами, вызывающая процедура начинает ожидание результатов выполнения вызываемой процедуры в момент вызова, а заканчивает в момент завершения вызываемой процедуры. Вместо этого предлагается начинать ожидание в момент, когда данные, вычисляемые вызываемой процедурой, требуются для дальнейших вычислений (если в этот момент они уже вычислены, то начинать ожидание не нужно), и заканчивать ожидание в момент, когда эти данные получены. Этот принцип иллюстрирует рис. 3.

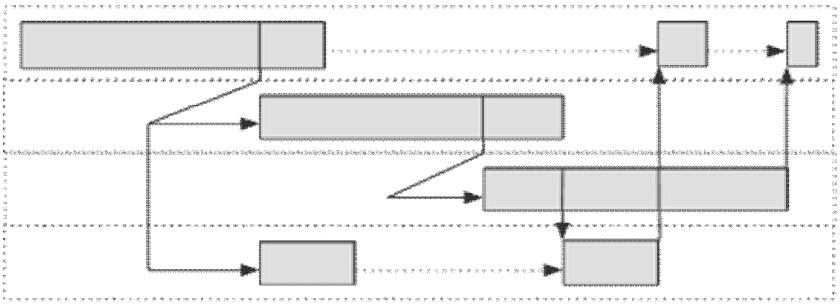

Рис. 3. Иллюстрация второго принципа технологии распараллеливания

Легко видеть, что такой график выполнения может быть получен из предыдущего максимально возможным сдвигом влево всех вычислений, при котором соблюдается следующее условие: всякое значение используется уже после того, как оно вычислено.

Архитектура программного обеспечения

Предлагаемая технология реализована в виде инструментальных программных средств на языке программирования Java [13]. Архитектура программного обеспечения приведена на рис. 4.

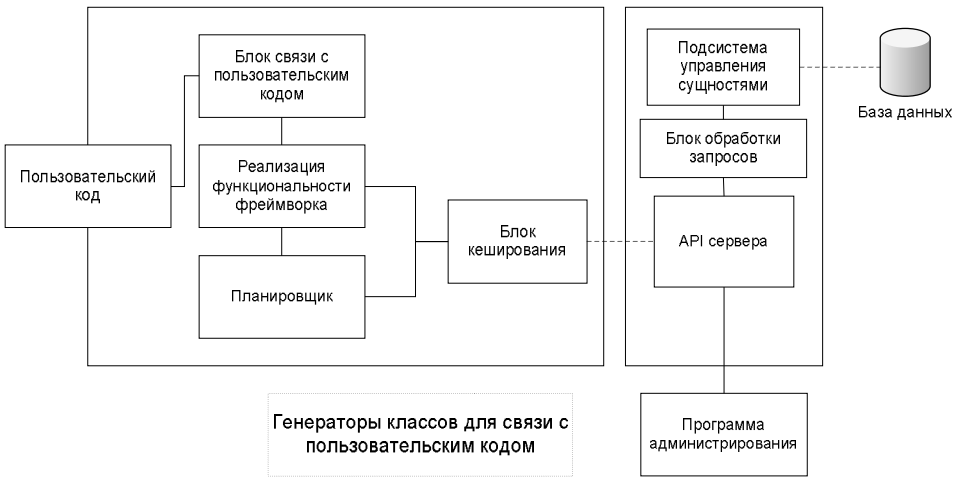

Рис. 4. Архитектура программного обеспечения

Серверная часть должна обеспечивать выполнение по запросам от клиентов следующих функций: отправки заказа, получения заказа, отправки значения, получения значения и отправки уведомления о завершении выполнения заказа. Также для отладки параллельных приложений к набору этих функций добавлены функции для получения текущего времени на сервере и вывода отладочного сообщения в создаваемый на сервере файл. Базовыми понятиями, возникающими в процессе взаимодействия клиента и сервера, являются заказ, идентификатор значения и идентификатор неизвестного значения.

Задача клиентской части – получение заказов с сервера и их выполнение, но также в состав этой части включены средства для отправки пользовательских заказов. Клиентская часть должна предоставлять пользовательскому коду функции отправки

заказа и получения значения величины; именно эти функции реализует часть клиента, ответственная за взаимодействие с пользовательским кодом. Следует пояснить наличие блока кэширования в клиентской части, введение которого связано с необходимостью снижения нагрузки на сеть. Этот блок хранит в себе полученные от сервера значения для некоторых идентификаторов значений и часть информации о связях между идентификаторами значений и идентификаторами неизвестных значений, а также о связях между идентификаторами неизвестных значений. Это позволяет не производить повторное получение данных, если они уже были недавно получены.

Функциональность, доступная пользователю, заключена в нескольких классах. Во-первых, это генерируемый класс библиотеки, который для каждой из выделенных пользователем процедур содержит метод для отправки заказа на выполнение этой процедуры. Список выделенных процедур при этом должен декларироваться пользователем в .XML файле, причем для каждой процедуры должно быть задано ее расположение, имя генерируемого метода, а также для каждого параметра – тип, имя и направление. Во-вторых, это класс ServConnector, содержащий средства для работы с расположенными на сервере файлами и добавления записей в лог-файл, ведущийся на сервере. В-третьих, это класс DataHandler, от которого должны быть порождены все классы, использующиеся в качестве типов параметров процедур. Пользователю доступны две группы методов этого класса:

- Абстрактные методы для сериализации и восстановления состояния. Для передачи параметров процедур не используется стандартный механизм сериализации, поскольку при передаче выходных параметров их значения нужно вносить в уже существующие объекты, в то время как стандартных механизм позволяет лишь создать новый объект. Поэтому реализуемые пользователем классы должны реализовывать эти методы.
- Методы preRead(), preChange(), preReplace(), которые должны вызываться перед обращением к данным, частичным изменением данных и полной заменой содержимого объекта соответственно. Для потомков этого класса вводится правило, что они должны предоставлять доступ к содержащимся в них данным только через свои методы, в которых вызываются эти три унаследованных метода, где это необходимо. Методы preRead и preChange убеждаются, что в объекте есть данные, а preReplace сбрасывает флаг о том, что в объекте нет данных, если он был установлен. Введение двух методов preRead и preChange позволяет, если после сериализации данных объекта производился только доступ на чтение, обнаруживать эту ситуацию и не производить второй раз сериализацию.

Выводы

Предлагаемый подход, в отличие от технологии MPI, предполагает, что программа, реализованная с его использованием, является набором инструкций, управляющим работой кластера в целом, а не его отдельного компьютера. Из этого следует применимость предложенного подхода: это алгоритмы, естественно реализуемые с использованием параллелизма заданий. Учитывая сложности с использованием технологии MPI в применении к параллелизму заданий, предложенная технология в этой области может рассматриваться как альтернатива традиционно используемой технологии MPI.

Список литературы

1. Антонов, А.С. Параллельное программирование с использованием технологии MPI / А.С. Антонов– М.: Изд–во МГУ, 2004. – 71 с.

2. Воеводин, В.В. Параллельные вычисления / В. В. Воеводин, Вл. В. Воеводин. – СПб.: БХВ-Петербург, 2002. – 608 с.
3. Gebali, F. Algorithms and Parallel Computing / F. Gebali – Publisher: Wiley, 2012. – 364 p. - ISBN-10: 0470902108 -ISBN-13: 978-0470902103
4. Buyya, R. Cloud Computing Principles and Paradigms / R. Buyya, J. Broberg, A. M. Goscinski. Publisher: Wiley, 2011. – 664 p. - ISBN:0470887990
5. Kolding, T. E. High-order Volterra Series Analysis Using Parallel Computing / T. E. Kolding, T. Larsen // International Journal of Circuit Theory and Applications. – 1997. - 25, 2. - 107-114 p. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.46.7254&rep=rep1&type=pdf
6. Pavlenko, V.D. Interpolation Method of Nonlinear Dynamical Systems Identification Based on Volterra Model in Frequency Domain / V.D. Pavlenko, S.V. Pavlenko, V.O. Speranskyy // Proceedings of the 7th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS'2013), 15–17 September 2013, Berlin, Germany – P. 173–178.
7. Pavlenko, V. Identification of systems using Volterra model in time and frequency domain / V.D. Pavlenko, S.V. Pavlenko, V.O. Speranskyy // In book: «Advanced Data Acquisition and Intelligent Data Processing» / V. Haasz and K. Madani (Eds.) – River Publishers, 2014 - P. 233-270.
8. Korbicz, J. Modeling, Diagnostics and Process Control: Implementation in the DiaSter System/ J. Korbicz, J. M. Kościelny – Berlin: Springer, 2010. – 384 p.
9. Pavlenko, V. Technology for Data Acquisition in Diagnosis Processes By Means of the Identification Using Models Volterra / V. Pavlenko, O. Fomin, V. Ilyin // Proc. of the 5th IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS'2009), Rende (Cosenza), Italy, September 21–23, 2009. – P. 327-332.
10. Тхань, Ф.Н. Параллельная реализация алгоритмов фильтрации космических изображений / Фыонг Нгуен Тхань, А.Ю. Шелестов // Проблемы управления и информатики. – 2005. – № 5. – С. 121 – 132.
11. Шпаковский, Г.И. Реализация параллельных вычислений: кластеры, многоядерные процесры, грид, квантовые компьютеры. – Минск : БГУ, 2010. – 155 с.
12. Основные классы современных параллельных компьютеров / Internet–ресурс: http://parallel.ru/computers/classes.html
13. Burdejnyj, V.V. Method of practical usage of orders based transparent parallelizing technology / V.V. Burdejnyj, V. D. Pavlenko // Труды Одесск. политехн. ун–та. – Одесса, 2012. – Вып. 1 (37). – С. 227-233.

ТЕХНОЛОГІЯ ПРОГРАМУВАННЯ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ НА КЛАСТЕРАХ

В.Д. Павленко

Одеський національний політехнічний університет,
пр. Шевченко, 1, Одеса, 65044, Україна; e-mail: pavlenko_vitalij@mail.ru

Розглядаються принципи організації кластерних обчислень за допомогою неявного розпаралелювання, заснованого на понятті замовлення, а також особливості нової, альтернативної MPI, технології розпаралелювання обчислень, що дозволяє для досить широкого класу обчислювальних алгоритмів природним чином переходити від існуючих послідовних реалізацій до паралельних.
Ключові слова: паралельні обчислення, кластери, технологія програмування паралельних обчислень, архітектура програмного забезпечення

TECHNOLOGY PROGRAMMING OF PARALLEL COMPUTING FOR CLUSTERS

V.D. Pavlenko

Odessa National Polytechnic University,
1 Shevchenko Ave., Odessa, 65044, Ukraine; e-mail: pavlenko_vitalij@mail.ru

Principles of organization of the cluster computing using implicit parallelism based on the notion of order, and also features a new, alternative MPI, parallel computing technology, allowing for a fairly wide class of computational algorithms in a natural way to move from existing serial to parallel implementations.
Keywords: parallel computing, clusters, technology programming parallel computing, software architecture