РЕАЛИЗАЦИЯ МЕТОДА ЧАСТИЦ-В-ЯЧЕЙКАХ В СИСТЕМЕ ФРАГМЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ LUNA

В.Э. Малышкин, В.А. Перепелкин

Институт вычислительной математики и математической геофизики СО РАН, г. Новосибирск

1. Введение

Система программирования LuNA предназначена для автоматизации генерации параллельных программ, реализующих большие задачи численного моделирования. Эта система разрабатывается в ИВМиМГ СО РАН [14]. LuNA-программа собирается из фрагментов данных и вычислений, т.е. из готовых модулей. Идея фрагментации данных и алгоритмов используется в программировании в течении ряда лет [1-6]. Разные модификации этого подхода использовались в системах программирования [2-5]. Большинство систем программирования использовали исполнительные системы (run-time системы) для организации вычислений [7-13].

В системе Charm++ [4] программа собирается из модулей (Chare). При этом представление алгоритма неотделимо от частичного распределения ресурсов. Это существенно снижает переносимость и настраиваемость программы Charm++. Также, в системе Charm++ нет средств для задания распределения данных и управления.

В PaRSEC [5, 6] данные назначаются на конкретные вычислительные узлы, что приводит к уменьшению переносимости. Также, возможность изменения интерпретации операций ограничена. Эти свойства, естественно, наследуются библиотекой D-PLASMA.

В [2], вместо обычно используемой исполнительной системы, были разработаны специализированное аппаратное обеспечение и операционная система, организующие исполнение программ.

В отличие от специализированных пакетов численного моделирования, таких как Fluent, Gaussian, и т. п., в системах программирования, таких как LuNA, конкретные прикладные численные алгоритмы программируются пользователем самостоятельно, а не используются готовые модели и алгоритмы. Вследствие этого, во-первых, пользователь не ограничен использованием заранее заложенных в пакеты моделирования алгоритмов и моделей, а, во-вторых, может «доверять» результату, выдаваемому прогаммой в том смысле, что точно известно, что делает программа, так как её писал сам пользователь.

2. Подход к конструированию программ в системе LuNA

Система фрагментированного программирования LuNA [14] нацелена на разработку прикладного программного обеспечения, реализующего большие численные модели. LuNA-программа собирается из фрагментов данных и вычислений, т. е. готовых модулей. Для того, чтобы достичь высокой производительности при исполнении прикладной LuNA-программы, системное программное обеспечение LuNA использует такие свойства численных алгоритмов как регулярность вычислений, данных и коммуникаций, существование сеток, отношение соседства как на данных, так и на операциях над этими данными. В статье преимущества подхода LuNA демонстрируются на примере программирования метода частиц-в-ячейках применительно к астрофизической задаче моделирования эволюции протопланетного диска [15].

3. Фрагментация метода частиц-в-ячейках

В [19] метод частиц-в-ячейках был приложен для моделирования переноса энергии в облаке плазмы. Там же рассматривается фрагментация алгоритмов метода. В описанной модели плазмы имеется два типа данных: 3D-решётка (пространство моделирования) и частицы. Каждая частица в каждый момент времени принадлежит конкретной ячейке. В ходе моделирования частицы перемещаются под влиянием электромагнитных сил, дискретизуемых на 3D-сетке и мигрируют из одной ячейки в другую. Обработка частиц состоит в вычислении её новых координат и её влияния на значения электромагнитного поля.

Кратко рассмотрим ключевые особенности алгоритма, существенные с точки зрения его фрагментации. Во-первых, вычисления являются локальными. Это означает, что для вычисления обновлённых сеточных значений используются значения лишь соседних узлов сетки, для расчёта сдвига частиц используются значения сетки только в той ячейке, в которой частица расположена, а сдвиг частицы осуществляется не далее чем на одну ячейку за один временной шаг. Исключение составляет глобальная коллективная операция редукции, применяемая на каждой итерации решения уравнения Пуассона. Во-вторых, уравнение Пуассона решается на уровне фрагментов явным методом, а внутри фрагментов — неявным (подробнее – см. [19]).

Фрагментация данных выполняется путём декомпозиции сетки на прямоугольные блоки – слои или ряды параллелепипедов ячеек. Блоки обрабатываются независимо, в частности, параллельно. Обработка блока состоит в обработке всех частиц, принадлежащих этому блоку и обработке сеточных значений

(дискретизованных полей). Объем вычислений (по времени) блока зависит от числа частиц, расположенных в нем. При расчёте исходно блоки распределены по процессорных элементам (ПЭ) мультикомпьютера таким образом, чтобы каждый ПЭ был загружен равномерно.

Так как частицы постоянно мигрируют, исходно имеющийся баланс нагрузки на ПЭ нарушается. Автоматическая динамическая балансировка нагрузки на ПЭ является основной проблемой параллельной реализации метода частиц-в-ячейках.

Если вместо электромагнитного используется гравитационное поле, фрагментация алгоритмов метода частиц-в-ячейках выполняется аналогичным образом.

4. Программирование метода частиц-в-ячейках в системе LuNA

Астрофизическая модель [15] моделирования эволюции протопланетного диска на базе метода частиц в ячейках была запрограммирована в системе LuNA. Было исследовано качество реализации этой модели. Тестирование показало, что накладные расходы системы LuNA не слишком высокие, динамические свойства прикладной программы обеспечиваются динамически, программирование модели не требовало хорошей подготовки в области параллельного программирования.

В соответствии с подходом системы LuNA, это приложение метода частиц-в-ячейках было запрограммировано в виде множества фрагментов данных и вычислений. Фрагменты данных описываются тут как параллелепипеды ячеек. Число ячеек, составляющих параллелепипед, является параметром программы, его можно изменить перед выполнением программы. Последовательные подпрограммы, обрабатывающие сеточные значения и частицы, должны быть разработаны пользователем.

На множестве фрагментов данных определяется отношение соседства. Все параллелепипеды, соседние с некоторым фрагментом данных, считаются соседними (как правило, это 26 соседей).

Фрагменты данных являются гранулой параллелизма. Они распределяются по ПЭ с учётом отношения соседства, то есть соседние фрагменты доллжны располагаться в одном ПЭ, или в ПЭ, непосредственно связанных сетью. В зависимости от топологии сети вычислителя учёт отношения соседства может быть выполнен с различным успехом. От качества такого распределения зависит нагрузка на коммуникационную подсистему вычислителя и, соответственно, на время выполнения программы. Распределение фрагментов по узлам выполняется автоматически системой LuNA. Более детально алгоритмы системы LuNA рассматриваются в [14].

В процессе исполнения фрагментированной программы динамическая балансировка нагрузки на ПЭ осуществляется специальным модулем. Этот модуль реализует алгоритм диффузионной балансировки нагрузки на ПЭ путём передачи фрагментов данных на соседние ПЭ, сохраняя при этом отношение соседства, то есть соседние фрагменты данных располагаются, по возможности, на соседних ПЭ. Этот модуль используется также для конструирования начального распределения нагрузки на ПЭ. Фрагменты данных фрагментированной программы загружаются в конкретные ПЭ, а затем они динамически балансируются с сохранением отношения соседства до тех пор, пока не будет достигнута равномерная нагрузка на ПЭ.

5. Тестирование системы LuNA

Для демонстрации способности системы LuNA эффективно исполнять фрагментированные программы в системе LuNA было запрограммировано и протестировано приложение метода частиц-в-ячейках для моделирования астрофизической задачи эволюции пылевого диска [15].

Общие условия тестов. Пространство моделирования представлено 3D-кубом из 64x64x64 ячеек (рис. 1). В центральной области находится пылевой диск, диаметр которого в 4 раза меньше, чем длина ребра куба. Частицы пыли вращаются вокруг центра масс с различными скоростями. Исходно неравномерное распределение частиц диска по ПЭ должно изменяться в ходе моделирования с целью выравнивания нагрузки на ПЭ.

Рис. 1

Исходно равное число фрагментов данных (параллелепипедов, состоящих из 4x4x4 ячеек) назначается на каждый ПЭ. Число фрагментов данных равно 16x16x16. Число частиц равно 10^7 и более. Как следствие исходно неравномерного распределения частиц по пространству моделирования распределение нагрузки на ПЭ также будет неравномерным, и этот дисбаланс должен быть выровнен в течение нескольких итераций. Для проверки правильности работы программы результаты моделирования сравнивались с результатами работы программы, написанной вручную [15].

Тест 1. Динамическая балансировка нагрузки. Цель этого теста – продемонстрировать насколько эффективно работает алгоритм динамической балансировки нагрузки на ПЭ. Выполнялись тестовые запуски с балансировкой нагрузки и без неё. Нагрузка на ПЭ измерялась каждые 10 с. Вариант с балансировкой нагрузки должен существенно снизить время моделирования.

Рис. 2

Вариант 1. Выполнение программы без балансировки (рис. 2). На рисунке 2 по оси X отложено время, ось Y соответствует номерам ПЭ, а по оси Z отложена нагрузка на процессор в процентах. Буквами A на рисунке обозначена обработка частиц, буквами B – решение уравнения Пуассона, а буквами C – простаивающие ПЭ. Виден дисбаланс нагрузки на ПЭ и этот дисбаланс сохраняется до завершения моделирования. Эффективность исполнения программы низкая, потому что только ПЭ с номерами 34 и 35 работают интенсивно, остальные ПЭ практически простаивают. Это связано с тем, что нагрузка на ПЭ пропорциональна числу частиц, в нем расположенных. Общее время моделирования составило 2320 с.

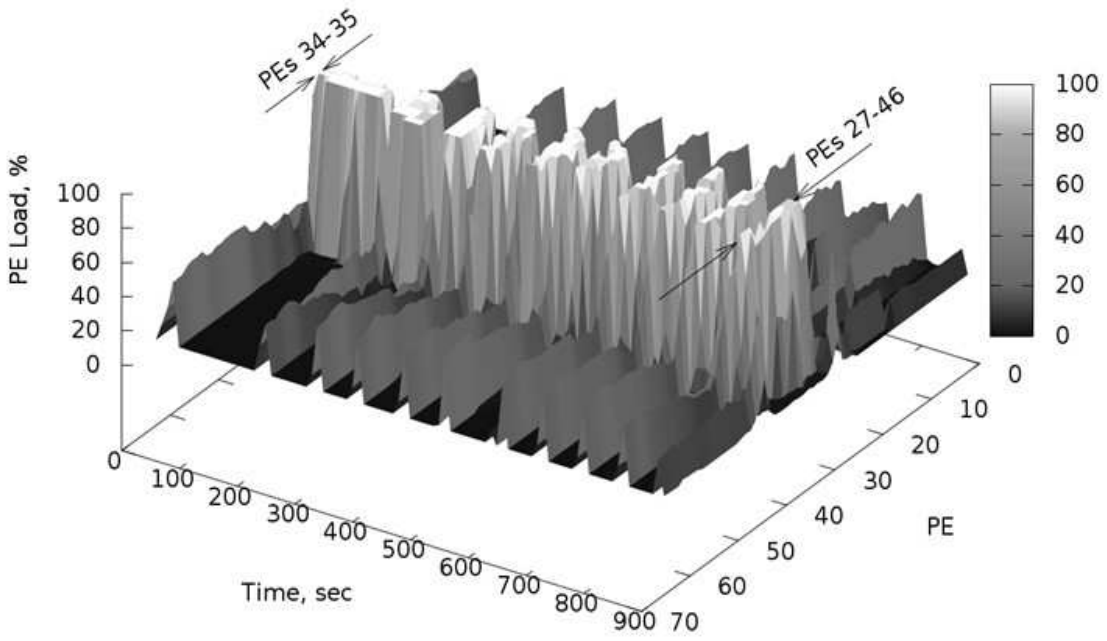

Рис. 3.

Вариант 2. Выполнение программы с балансировкой нагрузки (рис. 3). Был проведён тот же тест, при этом динамическая балансировка нагрузки на ПЭ была включена. На 800-й секунде от начала исполнения программы ПЭ с 27 по 46 вовлечены в интенсивные вычисления. Время расчёта одного временного шага сократилось, так как частицы распределились по ПЭ более равномерно. Общее время выполнения программы составило 860 с, то есть было уменьшено почти в 3 раза. Можно заметить, что число ПЭ, активно вовлечённых в вычисления в конце программы составило 19, т. е. можно было бы ожидать сокращения времени выполнения программы в 19/2=9,5 раз, а не в 3. Для этого можно назвать две основные причины. Во-первых, работа самого алгоритма динамической балансировки расходует часть процессорного времени и, что более существенно, нагружает коммуникационную подсистему вычислителя. Во-вторых, миграция фрагментов с одного ПЭ на другой имеет свою цену.

Тест 2. Оценка накладных системных расходов системы LuNA. Число фрагментов данных было зафиксировано, таким образом, и объем системных накладных расходов был фиксированным. Число частиц варьировалось от 10^6 до 10^9, и соответственно варьировался объем нагрузки, связанный с обсчётом частиц. Параметры модели были выбраны таким образом, чтобы общее время вычислений складывалось из системных накладных расходов и обсчёта частиц. На основе этих данных рассчитывалась доля системных накладных расходов при выполнении программы. На рис. 4 прямая показывает расчётное время выполнения программы в отсутствие накладных расходов. Кривая отображает результаты тестирования. Видно, что накладные расходы составляют незначительную долю вычислений, если число частиц равно 10^8 и более. Для численных моделей на основе метода частиц-в-ячейках это небольшое число частиц.

Рис. 4.

Тест 3. Оптимизация времени выполнения фрагментированной программы. Если в каждом ПЭ располагается не более одного фрагмента данных, то накладные расходы исполнительной системы LuNA близки к нулю. Большее число фрагментов данных в узле означает больший объём накладных расходов, но при этом возможность многопоточной обработки и сокрытия коммуникаций на фоне вычислений. Интерес представляет вопрос – какую долю накладных расходов следует считать приемлемой? Какой размер фрагментов данных обеспечит приемлемые накладные расходы? Данный тест призван ответить на эти вопросы.

Размер задачи фиксирован. Размер (и, соответственно, количество) фрагментов данных варьируется. Чем больше размер фрагмента данных, тем меньше накладных расходов. Программа исполнялась с различными размерами фрагментов данных, и измерялось время выполнения программы. Размер каждого фрагмента данных указан в ячейках по одной оси. Рисунок 5 показывает время выполнения программы в зависимости от размера фрагментов данных (например, размер 4 означает 3D-куб размера 4x4x4 ячейки).

Рис. 5.

По мере того как размер фрагментов данных увеличивается, время выполнения сначала сокращается, так как вместе с уменьшением количества фрагментов данных сокращаются и системные накладные расходы. Далее время начинает возрастать, потому что становится слишком мало фрагментов данных, чтобы обеспечить нагрузкой все ПЭ, что приводит к их простою.

6 Заключение

Система фрагментированного программирования LuNA автоматически обеспечивает все необходимые динамические свойства прикладной программы, такие как динамическая балансировка нагрузки на процессоры, настраиваемость, переносимость и т.д. Система обеспечивает высокую производительность прикладной программы и высокий уровень программирования численных приложений. Алгоритмы реализации системы LuNA нацелены на параллельную реализацию больших численных моделей на суперкомпьютерах.

7 Благодарности

Работа выполнена при финансовой поддержке РФФИ в рамках научных проектов № 14-01-31328 и № 14-07-00381.

ЛИТЕРАТУРА:

1. В.А. Вальковский, В.Э. Малышкин (1998) Синтез параллельных программ на вычислительных моделях. Наука, Новосибирск.
2. В.А. Торгашев, И.В. Царев (2001) Средства организации параллельных вычислений и программирования в мультипроцессорах с динамической архитектурой. Программирование, №4, с. 53-67.
3. Cell Superscalar, http://www.bsc.es/cellsuperscalar. Accessed 25 December 2013
4. Charm++, http://charm.cs.uiuc.edu. 25 December 2013
5. The Parallel Linear Algebra for Scalable Multi-core Architectures (PLASMA) project http://icl.cs.utk.edu/plasma. Accessed 25 November 2013
6. G. Bosilca, A. Bouteiller, A. Danalis, M. Faverge, T. Herault, J. Dongarra "PaRSEC: Exploiting. Heterogeneity to Enhance Scalability," IEEE Computing in Science and Engineering, Vol. 15, No. 6, 36-45, November, 2013.
7. W. Shu, L.V. Kale (1991) Chare Kernel – a Runtime Support System for Parallel Computations. J Parallel Distrib Comput, Vol. 11, Issue 3, pp. 198–211
8. K.V. Kalgin, V.E. Malyskin, S.P. Nechaev, G.A. Tschukin (2007) Runtime System for Parallel Execution of Fragmented Subroutines. 9th Int Conf Parallel Comput Technol, Springer Verlag, LNCS, Vol. 4671, pp. 544–552

9. R.D. Blumofe, C.F. Joerg, B.C. Kuszmaul, C.E. Leiserson, K.H. Randall, Y. Zhou (1995) Cilk: An Efficient Multithreaded Runtime System. ACM SIGPLAN Not, Vol. 30, Issue 8, pp. 207–216
10. I. Foster, C. Kesselman, S. Tuecke (1998) Nexus: Runtime Support for Task-Parallel Programming Languages. Cluster Computing, Issue 1(1), pp. 95–107
11. A.A. Chien, V. Karamcheti, J. Plevyak (1993) The Concert System – Compiler and Runtime Support for Efficient, Fine-Grained Concurrent Object-Oriented Programs. UIUC DCS Tech Rep R-93-1815
12. A.S. Grimshaw, J.B. Weissman, W.T. Strayer (1996) Portable Run-Time Support for Dynamic Object-Oriented Parallel Processing. ACM Trans Comput Syst (TOCS), Vol. 14, Issue 2, pp. 139–170
13. G.D. Benson, R.A. Olsson (1997) A Portable Run-Time System for the SR Concurrent Programming Language. Workshop Run-Time Syst Parallel Program (RTSPP)
14. V. Malyshkin, V. Perepelkin. Optimization methods of parallel execution of numerical programs in the LuNA fragmented programming system // The Journal of Supercomputing, Springer, Volume 61, Number 1 (2012), pp. 235-248.
15. S.E. Kireev. A Parallel 3D Code for Simulation of Self-gravitating Gas-Dust Systems // PaCT-2009 proceedings, Springer, LNCS 5698 (2009), pp. 406–413 (http://ssd.sscc.ru/en/dlb)
16. A.I. Maltsev. Algorithms and recursive functions / A.I. Maltsev; Translated from the first Russian ed. by Leo F. Boron, with the collaboration of Luis E.Sanchis, John Stillwell and Kiyoshi Iseki. - Groningen: Wolters-Noordhoff Pub. Co. - [1970]. - 372 p.
17. H. Rogers, Jr., 1967. The Theory of Recursive Functions and Effective Computability, second edition 1987, MIT Press. ISBN 0-262-68052-1 (paperback), ISBN 0-07-053522-1
18. S. Kireev, V. Malyshkin Fragmentation of numerical algorithms for parallel subroutines library - The J. of Supercomputing, Springer, Springer, Volume 57, Number 2 / August 2011 pp. 161-171
19. M.A. Kraeva, V.E. Malyshkin (2001) Assembly Technology for Parallel Realization of Numerical Models on MIMD-Multicomputers. Int J Future Gener Comput Syst, Elsevier Science. Vol. 17, No. 6, pp. 755–765
20. M. Gorodnichev, S. Kireev, V. Malyshkin. Optimization of inter-cluster communications in the NumGRID. – In the Proceedings of the second Russia-Taiwan symposium on Method and tools of Parallel Programming Multicomputers (MTPP). Springer, LNCS series, Vol. 6083, pp. 78-85, 2010
21. V.E. Malyshkin, V.A. Perepelkin. LuNA Fragmented Programming System, Main Functions and Peculiarities of Run-Time Subsystem // Proceedings of the 11th International Conference on Parallel Computing Technologies, LNCS 6873. - pp. 53-61, Springer, 2011.