Средства для задания императивного управления во фрагментированных программах на примере задачи моделирования самогравитирующего вещества методом частиц-в-ячейках

А.А. Ткачёва

Институт вычислительной математики и математической геофизики СО РАН, Новосибирск

В Лаборатории синтеза параллельных программ ИВМ и МГ СО РАН разрабатывается система фрагментированного программирования LuNA [1]. Технология фрагментированного программирования предназначена для реализации больших численных задач на суперкомпьютерах. В ней прикладная программа собирается из множества фрагментов, и фрагментированная структура программы сохраняется в ходе вычислений, что позволяет автоматизированно обеспечивать динамические свойства программы (динамическая балансировка загрузки процессоров, реализация коммуникаций на фоне счета, и т.д.). Такой подход к распараллеливанию позволяет программировать на более высоком уровне и уйти от некоторых сложностей системного параллельного программирования. Описание фрагментированного алгоритма является платформенно независимым, а настройку на конкретный вычислитель обеспечивает Runtime-система LuNA.

Был разработан модуль императивного управления (далее – прямого управления), его задачей является эффективная реализация фрагментированных подпрограмм. Главной чертой этого модуля является то, что он выполняет фрагментированную программу не по базовому алгоритму Runtime-системы LuNA, а в соответствии с дополнительно определенным прямым управлением. Кроме того, были разработаны средства представления прямого управления во фрагментированной программе.

Фрагментированная программа (ФП)

ФП представляет собой набор <*DF, CF, in, out*>, где *DF* – это множество фрагментов данных (ФД) единственного присваивания; *CF* – это множество фрагментов вычислений (ФВ) единственного

срабатывания. Для каждого $a \epsilon CF$ определены набор входных ФД *in(a)* и набор выходных ФД *out(a)*.

Выполнение ФВ состоит в вычислении его выходных ФД из значений входных ФД. ФВ реализуется некоторой функцией без побочных эффектов. Выполнение ФП заключается в выполнении всех его ФВ. Порядок выполнения ФВ в ФП основан на информационных зависимостях между ФВ.

Семантика базового алгоритма выполнения ФП

1. Среди множества всех ФВ выбираются те, чьи значения входных ФД уже вычислены. Такие ФВ называются готовыми к выполнению.
2. Из множества готовых к выполнению ФВ выбирается один или несколько ФВ на выполнение.
3. По мере выполнения ФВ какие-то ФД получают значения. При этом у некоторых ФВ все входные ФД оказываются вычисленными, и такие ФВ переходят во множество готовых к выполнению.
4. Пункты 2–3 повторяются, пока множество ФВ, готовых к выполнению, не пусто и ни один ФВ не выполняется.

Заметим, что, так как в общем случае порядок выполнения ФВ не детерминирован, базовый алгоритм выполнения ФП требует много времени и ресурсов. Чтобы уменьшить степень недетерминизма выполнения ФП, можно разработать и ввести дополнительные средства для задания императивного управления выполнения в ФП, которые позволили бы задавать желаемый порядок выполнения ФВ, но при этом какая-то степень недетерминизма еще оставалась бы для возможности параллельного выполнения на мультикомпьютере и обеспечения динамических свойств выполнения программы.

Целью работы является разработка средств задания прямого управления во ФП для системы LuNA и реализация их в виде программного модуля.

При этом эти средства должны удовлетворять следующим требованиям:

- позволять описывать желаемый порядок выполнения ФВ и распределение ресурсов для некоторого достаточно большого класса численных алгоритмов;

- представление алгоритма должно допускать эффективную реализацию на мультикомпьютере.

Управляющая сеть Петри

В качестве средств для задания прямого управления была выбрана модификация сети Петри (далее – управляющая сеть Петри (УСП)), а в качестве класса численных алгоритмов были выбраны итерационные алгоритмы (явная разностная схема, итерационные методы решения СЛАУ и т.д.). Модификацией сети Петри является то, что после срабатывания перехода не обязательно, что в каждом выходном месте появится фишка, это было введено для более прямого соответствия модели ФП.

Интерпретация УСП

Будем интерпретировать УСП следующим образом: Срабатыванию перехода сопоставим вычисление некоторого ФВ. Месту соотнесем некоторый буфер в памяти. Присутствие фишки в месте означает, что в этом буфере находится значения некоторого ФД. Ориентированные дуги между переходами и местами и местами и переходами будут сопоставляться с соответствующими наборами in и out для каждого ФВ. Порядок выполнения ФВ определяется функционированием УСП.

Использование УСП для задания прямого управления во ФП без использования базового алгоритма позволит существенно сократить накладные расходы. В то же время с помощью УСП можно задавать не только порядок выполнения ФВ, но и частичное распределение ресурсов, при этом к одной и той же ФП могут применяться различные УСП в зависимости от характеристик вычислительной среды.

Реализация программного модуля, осуществляющего прямое управление

В ходе работы был реализован программный модуль RuSh (Runtime Shell), который обеспечивает параллельное выполнение в общей памяти ФП, в которой порядок исполнения ФВ задан УСП. Язык реализации C++, для параллельного выполнения в общей памяти использовалась библиотека POSIX Threads.

Особенности и ограничения реализации: в разработанном программном модуле поддерживается только безопасная УСП (не более одной фишки в месте). Это условие позволяет в реализации ограничиться одним буфером для обмена ФД.

Моделирование самогравитирующего вещества методом частиц-в-ячейках

Исследование производительности и применимости разработанного модуля и УСП проводилось на задаче численного моделирования самогравитирующего вещества методом частиц-в-ячейках. Подробно эта задача описана в работе [2], а здесь приведем основные этапы алгоритма.

Моделирование разбивается на временные итерации, а каждая временная итерация состоит из следующих этапов:

- расчет плотности;
- расчет потенциала (решение уравнение Пуассона);
- расчет скорости;
- сдвиг частиц.

Фрагментация метода частиц-в-ячейках основана на декомпозиции области. В рассмотренном случае пространственная сетка разбивалась вдоль оси *Ox*. Этапы каждой временной итерации рассчитываются разными ФВ. На этапе расчета потенциала при решении уравнения Пуассона необходимо осуществлять обмен теневыми гранями потенциала, а также актуальными значениями плотности для границ. Также на каждой итерации происходит сдвиг, что влечет необходимость обмена частицами между ФВ.

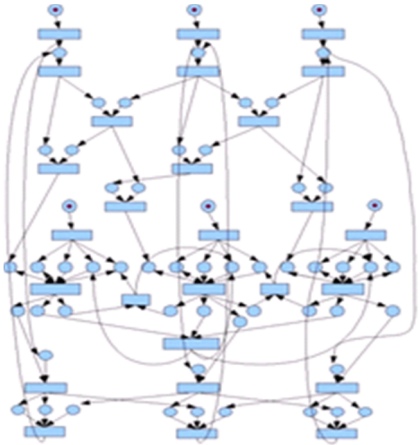

Рис. 1. УСП для метода частиц-в-ячейках при декомпозиции области на 3 подобласти

Для ФП метода частиц-в-ячейках была разработана УПС, показанная на рис.1. Заметим, что для хранения ФД различных временных итераций использовались одни и те же буферы памяти.

Исследование производительности и сравнительное тестирование

Исследование проводилось на 8-ядерном узле кластера, принадлежащего ССКЦ СО РАН (г. Новосибирск). Для тестирования метода частиц-в-ячейках использовалось два типа данных:

- данные А. Размер сетки 64x64x64, количество частиц 1000000, 1000 временных итераций моделирования;
- данные Б. Размер сетки 300x300x300, количество частиц 1000000, 100 временных итераций моделирования.

Такие параметры являются близкими к реально используемым [2] и поэтому позволяют судить о характеристиках программ в условиях, приближенных к реальным.

Тест 1. Сравнение с реализацией последовательного алгоритма

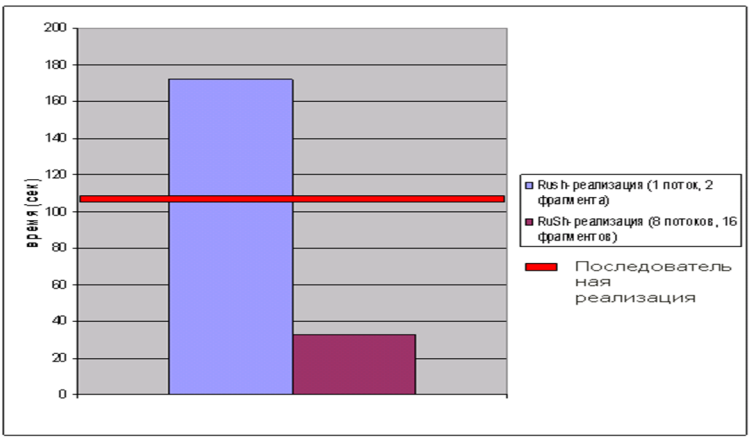

Рис.2. Последовательная реализация и реализации ФП в RuSh (данные А)

Целью данного теста было оценить накладные расходы, связанные с фрагментацией алгоритма, для этого было проведено сравнение времени работы реализации последовательного алгоритма и

реализации ФП в RuSh. Также было найдено оптимальное количество потоков и разбиение области, при котором достигается максимальное ускорение для оценки полученного выигрыша в производительности по сравнению с последовательным алгоритмом.

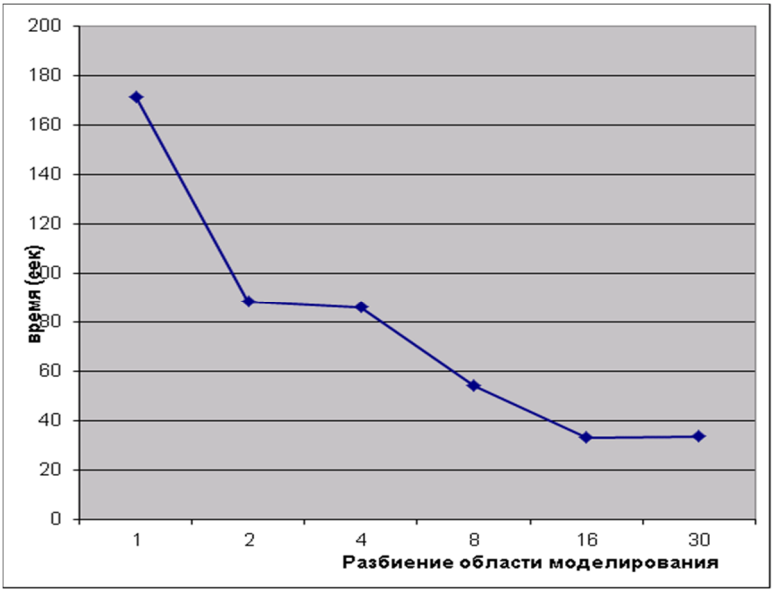

Рис. 3. Время работы Rush-реализации в зависимости от разбиения области моделирования для оптимального количества потоков (данные A)

На рис. 2 изображено время работы реализации ФП в RuSh при оптимальных параметрах. Ими оказались 8 потоков и декомпозиция области на 16 подобластей.

Результаты расчетов показывают, что реализация в RuSh на 1 потоке не сильно уступает реализации последовательного алгоритма. Накладные расходы, связанные с усложнением алгоритма после фрагментации, вполне терпимы. Параллельная реализация в RuSh работает быстрее, чем последовательная. При этом на 8-ядерном узле в идеале можно было ожидать 8-кратного ускорения. В нашем случае RuSh-реализация ускоряется максимум в 5,24 раза – это является хорошим результатом, так как решаемая задача относится к задачам с большим обращением к памяти, и

узким местом здесь выступает пропускная способность шины памяти.

Тест 2. Исследование масштабируемости

Под масштабируемостью в данном случае понимается способность системы справляться с увеличением рабочей нагрузки при добавлении ресурсов. Для её оценки сравнивалось время работы программы с увеличением степени декомпозиции области для оптимального количества потоков (рис. 3). Из рис. 3 видно, что реализация в RuSh справляется с нагрузкой, что говорит о её удовлетворительной масштабируемости при заданных параметрах.

Тест 3. Сравнительное тестирование с MPI-реализацией

Так как одним из требований к предлагаемым средствам для задания прямого управления во ФП является возможность эффективного исполнения на мультикомпьютере, то цель данного теста – провести сравнение производительности с реализацией на MPI, при этом для объективности сравнения все MPI процессы запускались на одном узле, чтобы свести к минимуму накладные расходы, связанные с доступом к памяти между процессами.
Из рис. 4 видно, что производительность RuSh-реализации практически совпадает с производительностью MPI-реализации, что

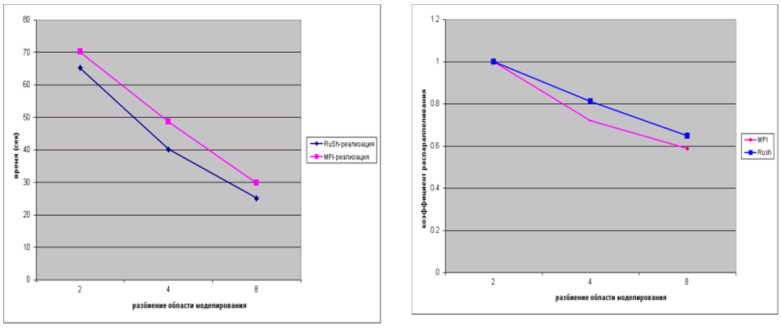

Рис.4. Сравнение времени работы и коэффициента распараллеливания Rush- и MPI- реализаций в зависимости от разбиения области моделирования (данные Б)

является хорошим результатом и говорит о том, что разработанный

модуль позволяет эффективное исполнение на мультикомпьютере ФП, в которой управление задано УСП.

Заключение

Были предложены средства для задания прямого управления при ФП для системы LuNA. Реализован программный модуль RuSh для параллельного выполнения на мультикомпьютерах с общей памятью ФП, в которой управление задано УСП.

Разработана УСП для задачи моделирования самогравитирующего вещества методом частиц-в-ячейках и реализована соответствующая ФП. Проведено сравнительное тестирование производительности между RuSh- и MPI-реализациями на многоядерном мультипроцессоре. Результаты сравнительного тестирования показали применимость предлагаемых средств.

Литература

1. *Malyshkin V.E., Perepelkin V.A.* LuNA Fragmented Programming System, Main Functions and Peculiarities of Run-Time Subsystem // Proceedings of the 11-th Conference on Parallel Computing Technologis, LNCS 6873. Springer, 2011. P. 53–61.
2. *Киреев С.Е.* Параллельная реализация метода частиц-в-ячейках для моделирования задач гравитационной космодинамики // Автометрия. 2006. Т.42, № 3. С.32–39.