УДК 519.6:004.94+004.414.23

А.В. ГАХОВ, В.О. МИЩЕНКО

Харьковский национальный университет им. В.Н. Каразина, Украина

СХЕМА ПАРАЛЛЕЛЬНОЙ МОДИФИКАЦИИ СИСТЕМ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ, ИСПОЛЬЗУЮЩИХ МЕТОДЫ ДИСКРЕТНЫХ ОСОБЕННОСТЕЙ

Стремление к улучшению качества программного обеспечения численного моделирования на основе методов дискретных особенностей (МДО) по характеристике «эффективность» заставляет обращаться к параллельным вычислениям, в частности – кластерным. Например, 3D модели дифракции волн на изолированных плоских экранах произвольной формы потребовали распределения вычислений по многим компьютерам, если оперативной памяти одного не хватает для загрузки матрицы модели. Недавно проблема усугубилась в результате создания моделей подобных процессов в слоисто-неоднородных средах, так как время расчета матрицы дискретной моделирующей системы также стало критическим фактором. Опыт частных приемов, помогавших улучшать подхарактеристики эффективности в частных задачах, позволяет в настоящее время предложить метод такого улучшения, который может быть востребован при решении других задач компьютерного моделирования с использованием МДО, столкнувшихся с проблемами ресурсной и временной эффективности.

Ключевые слова: *методы дискретных особенностей, численное моделирование, эффективность, дифракция, параллельные вычисления, кластер.*

Введение

Практические вопросы распараллеливания вычислений, включая вычислительные кластеры, порождают сложные задачи программной реализации алгоритмов [1]. Для численного моделирования по методам дискретных особенностей (МДО) [2, 3] повышение эффективности в смысле увеличения размерности решаемых задач за счет распараллеливания при тех же возможностях одного аппаратного вычислителя, весьма актуально [4].

Так, выполнение расчетов согласно 3D модели дифракции волн на изолированных плоских экранах произвольной формы при тогдашних параметрах типовых персональных компьютеров (ПК) не позволяло достичь необходимой степени точности в силу ограничений по памяти [4]. Преодолеть эту трудность нам впервые удалось в результате распределения работы с матрицей системы линейных алгебраических уравнений (СЛАУ) по более, чем десятку ПК в сети [5]. Реализованный алгоритм, по сути, был известным [6]. При этом выяснилось, что получить одновременно выигрыш в скорости нельзя, не удвоив число узлов и не улучшив прочие параметры импровизированного вычислительного кластера. Поэтому, как только на ПК стало доступно на порядок больше оперативной памяти, чем раньше, распределенная обработка перестала использоваться. Она снова понадобилась, когда перешли к моделированию дифракции на экранах, размещенных на границе полупространства из иного материала [7, 8]. Изменились и параметры общедоступных кластерных систем: обычно 4-ядерные процессоры в узлах и не менее, чем гигабитные соединения. Прежняя схема параллельной модификации оправдала себя в новых условиях, она допускает обобщение для применения в других задачах.

1. Цель и задачи исследования

Цель исследования – разработка и апробация метода модификации последовательных программ реализации вычислений МДО для их выполнения в вычислительном кластере при больших размерностях задач и как можно эффективнее по времени.

Главным было требование гибкости метода, чтобы он давал не единственную версию параллельной реализации, а позволял получать подходящие версии для новых постановок задачи и новых кластерных систем в процессе их совершенствования.

Описание метода требовалось в форме схемы действий реализаторов компьютерного моделирования по МДО и рекомендуемых схем программ.

2. Подготовка вычислений в кластере

МДО в математическом моделировании физических процессов приводит к таким алгоритмам,

как обработка массива элементов, при условии, что обработка одних частей массива зависит от результатов обработки некоторых других частей. В анализе дифракционных явлений примерами служат задачи генерации элементов СЛАУ и решение этой системы. Поскольку применение МДО всегда обеспечивает [2] матрицу СЛАУ с диагональным преобладанием в прямом или обобщенном смысле, обычно СЛАУ решают методом Гаусса без перестановок (строк, столбцов) [7, 8]. Компактная схема этого метода допускает, как известно, распараллеливание алгоритма [6]. Используя в алгоритмах МДО раздельно выполняемые вычислительные процессы, легко свести их взаимодействие к записи массивов («файлов») в память подходящего вида одними и чтения другими процессами. Память можно защитить мониторами, которые из современных индустриальных языков программирования явно обеспечивает Ада («защитные модули»). Взаимодействующие процессы в этом языке – «задачи» [9]. Задачи в Аде приспособлены к реализации алгоритмов, допускающих параллельное выполнение своих частей [10].

Дадим общую схему, подразумевая пример построения и решения СЛАУ. Иллюстрации даем на языке Ада, понятном и тому, кто знает Паскаль.

Пусть T – корневой модуль-пакет разрабатываемой программы. Интерфейсная часть пакета, обеспечивающего параллельные вычисления такова:

```
-- Комментарии начинаются двумя «минусами»
with T;     -- присоединить модуль описаний T
use T;      -- открыть прямую видимость в T
package Distribution is
    -- Описание необходимых технических средств:
    M: constant Machine_Index:= T.M;
    subtype M_Index is Index range 1..M;
    task type Processor (S: M_Index);
    -- это тип задачи-обработчика, который снабжен
    -- параметром S («дискриминантом»)
     task type Generator;
    -- тип задачи-генератора СЛАУ, который тоже
    -- часто имеет дискриминанты (параметры)
end Distribution;                          --        (1)
```

В теле этого пакета должны быть тела задач, реализующих алгоритмы генерации элементов обрабатываемого массива и алгоритмы самой обработки. Пример главной процедуры узла кластера:

```
with Distribution, Text_IO, Ada.Integer_Text_IO;
use Distribution, Text_IO, Ada.Integer_Text_IO;
procedure Parallel_ Processor is
    procedure Start (S: M_Index) is
        P: Processor (S);--смысл дискриминанта
        -- - это номер, передаваемый при вызове
    begin Put (" #"); Put (S, 1); New_Line;
    end Start;
begin
    Put ("Parallel Prog, ver.2. , Pr.");
     Start (S_Value); -- функция S_Value читает
    -- номер (в диапазоне 1, .. M) из ini-файла
end Parallel_Processor;                    --        (2)
```

При отладке на одном компьютере:

```
with Distribution, Text_IO,Ada.Integer_Text_IO;
use Distribution, Text_IO,Ada.Integer_Text_IO;
procedure Parallel_Program is
    procedure Pr_Start (S: M_Index) is
        P: Processor (S);
    begin Put (" #");Put (S, 1); New_Line;
    end Pr_Start;
    procedure Gn_Start (S: M_Index:=0) is
        P: Generator;
    begin Put (" #"); Put (S, 1); New_Line;
    end Gn_Start;
begin
    Put ("Parallel Prog, ver.1,");
    Gn_Start;
    for I in M_Index loop
        Start(I);
    end loop;
end Parallel_Program;                      --        (3)
```

Гипотеза об относительной корректности: на одной машине реализация каждого из задачных типов пакета Distribution позволяет безаварийно (но, быть может, с ошибками) рассчитывать все необходимые для межзадачного взаимодействия значения (передаваемые через файлы), если получаемые от других задач промежуточные дынные корректны.

По предположению, обработчик с индексом S = r , в определенный момент своей работы нуждается в получении данных из некоторого файла реализации взаимодействия f , который должен быть записан другим обработчиком или генератором. На случай, если к моменту обращения обработчика к файлу f этот файл не создан, тот ожидает, повторяя обращения каждые Δ секунд. При отладке на одном компьютере это при естественной реализации задач приводит к тупику. Проблему можно снять так.

Пусть F_0 – множество файлов реализации взаимодействия, которые создаются процессами без какого-либо использования других файлов. Пусть F_i – множество файлов реализации взаимодействия, создание которых требует только файлов множества F_{i-1}. Перед отладкой разместим в соответствующих областях памяти, если нужно, макетные файлы взаимодействия, эквивалентные реальным по типам и количествам записей. Следующее условие выполнимо для вычислительных схем МДО [2,7,8].

Постулат простой отладки. Объединение F_i покрывает все K файлов взаимодействия и можно избрать последовательность пусков программы с такими порядками запуска процессов, что всякий файл f из всякого F_i в результате всех выполнений станет реальным и заменит свой макет (пуски могут завершаться тупиками, но считываемые из макетов данные не приводят к аварийным завершениям).

Следствие. Пусть при реализации такой отладки после пуска программы часть созданных файлов ошибочны вследствие дефектов программы, которые после этого исправляются. Тогда для того, чтобы все макетные файлы заменились реальными, достаточно провести 2K отладочных пусков (возможно, с изменениями порядков запуска процессов).

3. Оптимизация кластерных вычислений

Если распределение вычислений на n узлов, решает проблему нехватки памяти, то ради ускорения счета можно варьировать количества машин под обработчики ($m \le n$) и под генераторы (1).

Оптимизация требует оценивать продолжительности вычислений по математической модели временных затрат, зная параметры алгоритмов, латентности и пропускные способности сети [1].

Для алгоритмов нужно определить из теории числительных методов показатели приближенно степенных зависимостей времени выполнения от размерности. Мультипликативные постоянные оцениваются по результатам экспериментальных прогонов программы.

Превратим в имитационную модель саму программу вычислений, вставляя в нужные места операции, моделирующие затраты времени в кластере. Эти операции описываются в следующем пакете:

```
with Virtual_System_Siz,; -- и необходимые пакеты;
use – необходимые пакеты;
package Simulation_of_Parallel_Execution is
  N: constant Positive:= Virtual_System_Size;
-- (N это размерность моделируемой СЛАУ)
  L: constant Natural:= Number_of_Generators;
  M: constant Positive:= Number_Of_Machines;
-- Опустим описания вспомогат. констант и типов.
  Simulated_Error: exception;
  subtype Comp is Integer range 1-L..M;  -- в этот тип
                -- преобразуются адреса файлов и задач
  type Name is    record S: Comp; -- и другие поля
                  end record;
  type Time_Interval is private;
  type Comps_List is private;
  type Files_Table is private;
  type Communic_List is private;
  type Communic_Ind is private;
  Null_Communic_List: constant Communic_List;
  Null_Ind: constant Communic_Ind;
  protected type Claster is --монитор сопрограмм
    -- Операции моделирования временных затрат:
    entry Time_Up_To (File_Work: Name);
    entry Try_To_Read (Comp) (What: Name);
    entry Work_For_Writing (Comp) (What: Name);
    -- Операции вывода временных диаграмм:
    procedure  Type_History_of (Pr: Comp; To:
                                        Info_File);
    procedure  Type_Time_Of (Set_of_Files: Id;
                                  To: Info_File);
  private
    entry Wait_For_Reading (Comp) (What: Name);
    -- это постановка в очередь задач-обработчиков
  --Таблицы для временных диаграмм:
    Processor: Comps_List;
    File_On: Files_Table;
    procedure  Init_Files_Table;--инициализатор
    Communications_Of: Communic_List:=
                          Null_Communic_List;
    Ind_Of: Communic_Ind:= Null_Ind;
  end Claster;
private
--Детали реализации private-типов.
end Simulation_of_Parallel_Execution;      --        (4)
```

«Главным» в пакете является защитный тип Claster, координирующий согласованную генерацию и обработку. Можно вводить инициализаторы:

```
package  Simulation_of_Parallel_Execution.On_1   is
  type Claster_1 is new Claster;
  Comp_Net: Claster_1;
private
  procedure Comp_Net_Init (Net: in out Claster_1);
-- это служебная операция.для Comp_Net
end Simulation_of_Parallel_Execution.On_1;--       (5)
```

Операции над файлами вроде *Read* при имитации меняются, и должны оформляться так:

```
with Simulation_of_Parallel_Execution.On_1;
 use Simulation_of_Parallel_Execution.On_1;
separate (T.S1.S2) – указан модуль-хозяин
  procedure  Read          -- терминальный субмодуль
    ...
  end Read;                                --        (6)
```

Теперь главная процедура определяет для модулей-задач режим сопрограмм:

```
with Simulation_of_Parallel_Execution.On_1;
use Simulation_of_Parallel_Execution.On_1;
with Distribution, Text_IO, Ada.Integer_Text_IO;
use Distribution, Text_IO, Ada.Integer_Text_IO;
procedure Simulation_Program is
    P0: Generator;
    P1: Processor (1); P2: Processor (2); -- ...
begin
    Put_Line("Parallel Prog, ver.3. ,");
```

```
    Comp_Net.Type_History_Of (0, "p00");
    Comp_Net.Type_History_Of (1, "p01");
    Comp_Net.Type_History_Of (2, "p02");
end Simulation_Program;                    --        (7)
```

Моделируемая размерность может быть сколь угодно большой, но имитация длится не больше 1 с.

4. Экспериментальная часть

Описанной метод сперва был испытан в задаче моделирования дифракции скалярных волн на изолированном плоском экране по МДО: генерация квадратной матрицы СЛАУ, ее LR-факторизация и решение СЛАУ для сгенерированной правой части. Пакет (1), названый SLAE_Distribution, поддерживал вариант параллельного алгоритма, основанного на формулах компактной схемы факторизации, в котором каждая задача-обработчик обслуживает часть прямого и обратного хода Гаусса, относящегося к одной из M горизонтальных полос матрицы СЛАУ. Роль корня T у нас играет пакет Matrix_Strips. Он определяет тип «полоса» (горизонтальная матрицы СЛАУ), а его в дочерние пакеты определяют типы «вертикальная полоса» и «блок» (соответствует пересечению «полосы» с «вертикальной полосой»). s -й обработчик обслуживает s -ю сверху полосу. Полоса разбивается на M квадратных блоков. Блоки исходной матрицы, вообще говоря, нужно читать из файлов, создаваемых генераторами, т.е. задачами типа SLAE_Generator. Для вычисления блоков s -х полос матричных множителей другие блоки треугольного R-множителя передаются через файлы, созданные обработчиками верхних полос. Тем самым, по отношению к решению СЛАУ постулат простой отладки выполняется.

Модули имитации вычислений в кластере (см. (4)-(5)) ориентированы на подсистему, развивающую модуль (1), содержащую самые разные модули работы с файлами взаимодействия. Это чтение и запись матричных колонок, «разбрасывание» их по разным машинам. Но трех операций моделирования временных затрат (см. (4)) оказалось достаточно.

Сравнение времен, полученных при имитации и выполнении параллельной программы в кластерах A (11×633МГц /128МБ /100Мб/с) и B (10×2,4ГГц /256МБ /100 Мб/с.) обнаружило такие погрешности:

до 1 % для A на примерах с размерностями СЛАУ не более 3600 при использовании 4-8 узлов, из котрых не более 2 генераторов;

до 15% для B на размерностях 4800-7200, 10 узлов (не более двух генераторов).

Затем метод испытывался в более сложной задаче 3D дифракции [8], где продолжительность расчета матрицы СЛАУ увеличивается так, что вместе с затратами на запись файлов взаимодействия он становится определяющим фактором. Тактика упростилась: продолжительность расчета заметно снижается с увеличением числа генераторов матрицы СЛАУ. Эксперименты были перенесены в мини-кластер C ((2×(4×2.2ГГц)+(3×2.3)) /3.7ГБ /1Гб/с), где есть возможность поместить в один узел и генератор, и обработчик, а файл матричных элементов реализовать в оперативной памяти. Подтвердилось, что при данной структуре программы все необходимые модификации осуществимы путем добавления модулей с незначительными изменениями в основной «параллельной» части программы. Такие модификации верифицируемы, отладка не требуется.

5. Анализ полученных результатов

Элементарные оценки показывают, что распределение исследуемых вычислений по не более, чем 11 узлам не позволяет решить практически значимую задачу увеличения степени дискретизации вдвое за счет распараллеливания. Но нами отлажена схема (1) – (7) и степень достижимой дискретизации поднята в 1.5 раза, что тоже важно. Памяти хватает, но требуемое время меряется сутками, так что эксперимент с предельной размерностью был проведен только однажды. Проверен эффект имитационного моделирования. С его помощью при N = 2400, 4800 минимизировалось время выполнения, варьируя l при $l+m=n_0$ ($n_0=10,11$) и Δ (цикл ожидания). Фактический результат в кластере был лучше или не хуже в сравнении с 8-ю другими вариантами, в которых менялось число генераторов (±1), и (или) цикл ожидания (в 0.5 или 1.5 раза).

Заключение

Разработана схема параллельных реализаций МДО и их отладки на одном компьютере с последующей механической модификацией кода для получения программ, распределяемых на узлы вычислительного кластера. Определены простые условия на характер алгоритма (в случае МДО они выполняются), гарантирующие завершение одномашинной отладки многозадачной программы за конечное время. Дана также схема модификации данной программы в средство имитации ее выполнения.

Значимый прикладной эффект для 3D МДО ожидается в кластерах, имеющих свыше 64 узлов.

Следует изучить возможность ускорения вычислений МДО на многоядерных узлах за счет улучшения использования промежуточной памяти.

Литература

1. Воеводин В.В. Параллельные вычисления / В.В. Воеводин, Вл.В. Воеводин. – СПб., 2002. – 608 с.

2. Белоцерковский С.М. Численные методы в сингулярных интегральных уравнениях / С.М. Белоцерковский, И.К. Лифанов. – М.: Наука, 1985. – 256 с.

3. Гандель Ю.В. Введение в методы вычисления сингулярных и гиперсингулярных интегралов. Учеб. пособие / Ю.В. Гандель. – Х: ХНУ, 2001. – 92 с.

4. Мищенко В.О. К моделированию электромагнитных явлений на базе использования методов дискретных особенностей для решения гиперсингулярных интегральных уравнений / В.О. Мищенко // Труды международной конференции по вычислительной математике МКВМ-2004. – Ч. II. – Новосибирск: Изд. ИВМиМГ РАН, 2004.– С. 555-560.

5. Mishchenko V.O. Ada programming language and specifying, modeling and distributed computing for the implementation of the discrete singularities methods / V.O. Mishchenko // Procedins of SCALNET-2004, KSPU. – Kremenchug, 2004. – P. 110-112.

6. Ортега Дж. Введение в параллельные и векторные методы решения линейных систем / Дж. Ортега. – М.:Мир, 1991. – 367 с.

7. Гандель Ю.В. Псевдодифференциальные уравнения электромагнитной дифракции на плоскопараллельной структуре и их дискретная модель / Ю.В. Гандель, В.О. Мищенко // Вісник Харківського нац. ун-ту. – № 733. – Х., 2006. – С. 58-75.

8. Гахов А.В. Трехмерная модель метода дискретных особенностей рассеяния скалярных волн экраном на границе раздела сред / А.В. Гахов, В.О. Мищенко // Вестник херсонского национального технического университета: Сб. науч. трудов. – №2(25). – Херсон, 2006 – С. 135–140.

9. Barnes J. Programming in Ada 2005 / J. Barnes – Addison Weslay, 2006. – 828 p.

10. Burns A. Concurrent and Real-Time Programming in ADA 2005 / A. Burns, A. Wellings. – Cambridge University Press, NY, USA 2007. – 461 p.

Поступила в редакцию 12.02.2010

Рецензент: д-р техн. наук, проф., зав. кафедрой информатики М.Л. Угрюмов, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков, Украина.

СХЕМА ПАРАЛЕЛЬНОЇ МОДИФІКАЦІЇ СИСТЕМ КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯ, ЩО ВИКОРИСТОВУЮТЬ МЕТОДИ ДИСКРЕТНИХ ОСОБЛИВОСТЕЙ

А.В. Гахов, В.О. Міщенко

Прагнення до покращення якості програмного забезпечення чисельного моделювання на основі методів дискретних особливостей (МДО) за характеристикою «ефективність», примушує звернутися до паралельних обчислень, зокрема – кластерних. Наприклад, 3D моделі дифракції хвиль на ізольованих плоских екранах довільної форми вимагали розподілення обчислень по багатьом комп'ютерам, якщо оперативної пам'яті одного не вистачає для завантаження матриці моделі. Нещодавно проблема поглибилася в результаті створення моделей подібних процесів у слоїсто-неоднорідних середовищах, тому що час розрахунку матриці дискретної системи моделювання також став критичним фактором. Досвід спеціальних прийомів, що допомагали поліпшити підхарактеристики ефективності в окремих задачах, зараз дозволяє запропонувати системний метод такого удосконалення, який може бути затребуваний при розв'язанні багатьох задач комп'ютерного моделювання з використанням МДО, у випадку наявності проблем ефективності за часом та ресурсами.

Ключові слова: методи дискретних особливостей, чисельне моделювання, ефективність, дифракція, паралельні обчислення, кластер.

PARALLEL MODIFICATION SCHEME FOR COMPUTER MODELING SYSTEMS THAT USE DISCRETE SINGULARITIES METHODS

A.V. Gakhov, V.O. Mishchenko

The improving of DSM software quality leads to a parallel computing, in particular, to computing on clusters. For instance, 3D models of wave diffraction on isolated arbitrary flat screens require the distributed computing if a memory is not enough to load the matrix of the model. Recently, a problem exacerbated by the developing of models of similar processes in layered media, since the calculation time of the matrix of the discrete system is a critical factor also. Experience of private receptions, that helped to improve the efficiency in special cases, allows us to propose a systematic method for such improvement. This method can be relevant in solving of many problems of computer simulation using MDS with problems of resources and time efficiency.

Keywords: discrete singularities methods, numerical simulation, efficiency, diffraction, parallel computing, cluster.

Гахов Андрей Владимирович – канд. физ.-мат. наук, доц. кафедры искусственного интеллекта и программного обеспечения, Харьковский национальный университет им. В.Н. Каразина, Украина, e-mail: gahov@univer.kharkov.ua.

Мищенко Виктор Олегович – канд. физ.-мат. наук, доц., доц. кафедры моделирования систем и технологий, Харьковский национальный университет им. В.Н. Каразина, Украина, e-mail: mischenko@univer.kharkov.ua.