УДК 681.3.06:004.4(045)

РЕАЛІЗАЦІЯ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ НА БАГАТОЯДЕРНИХ ПЕРСОНАЛЬНИХ КОМП'ЮТЕРАХ

Іванова Л. М., канд. техн. наук; Чистяков О. В., Ісламова І. С.

Національний авіаційний університет

ivanova@csfnau.kiev.ua

Проведено експериментальне дослідження застосування програмного засобу OpenMP для розпаралелювання програм на багатоядерних персональних комп'ютерах з метою підвищення їх продуктивності під час розв'язування задач великих розмірів.

Ключові слова: паралельне програмування, багатоядерний процесор, розпаралелювання, багатопоточність, інкрементне програмування, OpenMP.

Is conducted the experimental research of application of a software OpenMP for multisequencing the programs on multinuclear personal computers with the purpose of increase of their productivity at problem solving.

Key words: parallel programming, multicore processor, paralleling, multithreading, incremental programming, OpenMP.

Постановка проблеми

У більшості випадків математичного моделювання фізичних, хімічних, економічних, механічних і т. п. процесів виникають науково-технічні задачі великих розмірів, які потребують потужних обчислювальних ресурсів (пам'яті і швидкодії) сучасних комп'ютерів. Питання підвищення продуктивності комп'ютерів дуже актуальне. Нині одним з основних напрямів вирішення цього питання є створення паралельних комп'ютерів з розподіленою пам'яттю різних архітектур, наприклад, суперкомп'ютерів, систем кластерного типу тощо. Для створення програм з паралельною організацією обчислень для паралельних комп'ютерів з розподіленою пам'яттю використовують спеціальні засоби паралельного програмування такі, наприклад, як *MPI* (*Message Passing Interface*), *PVM* (*Parallel Virtual Machine*) [1]. За допомогою функцій цих програмних засобів виникають паралельні процеси, які одночасно виконуються на різних процесорах паралельного комп'ютера. Кожен процес працює зі своєю локальною оперативною пам'яттю. При цьому основним способом під-тримки взаємозв'язків між процесами є передача інформації та даних один одному.

Неухильно зростає продуктивність персональних комп'ютерів. Починаючи з 2005 року провідні фірми-виробники *Intel*, *AMD* і інші створюють дво- та чотириядерні персональні комп'ютери. Але, як показала практика, наявність багатоядерних процесорів у персональних комп'ютерах не дала очікуваного збільшення продуктивності в прикладному програмному забезпеченні. Основна проблема, з якою зіткнулися користувачі, — відсутність готового програмного продукту, функціонуючого в багатопотоковому режимі. На сьогодні до випуску готується шестиядерний процесор *Core i9*, а проблема ефективності використання таких процесорів залишається. За статистикою лише 30 % створеного програмного забезпечення враховує багатоядерність процесорів. Навіть спроби розпаралелювати потоки за допомогою операційної системи далеко не у всіх випадках призводять до позитивного результату.

На разі одним з найбільш популярних засобів паралельного програмування на багатоядерних комп'ютерах із загальною пам'яттю є програмний засіб *OpenMP*, за допомогою якого можна розпаралелити окремі частини програми, які будуть виконуватися на ядрах комп'ютера окремими потоками. При цьому не потрібно описувати обмін інформацією.

Аналіз досліджень і публікацій

У праці [1] розглянуто сучасні архітектури паралельних комп'ютерів та технології паралельного програмування, в тому числі при використанні програмного засобу *OpenMP*.

У працях [2; 3; 4; 5] описано основні директиви, функції та змінні оточення стандарту *OpenMP* з деякими прикладами їх застосування в програмах алгоритмічними мовами *Ci* та *Fortran* на багатопроцесорних системах з єдиною пам'яттю.

Конкретні приклади застосування програмного засобу *OpenMP* для реалізації розпаралелювання програм на багатоядерних персональних комп'ютерах не розглядаються.

Мета

Основна мета статті — дослідити підвищення продуктивності багатоядерних персональних комп'ютерів за рахунок розпаралелювання програм, що реалізують розв'язування задач великих розмірів. На конкретних прикладах показати як можна застосовувати OpenMP для розпаралелювання програм на багатоядерних персональних комп'ютерах та провести аналіз продуктивності багатоядерних персональних комп'ютерів.

© Л. М. Іванова, О. В. Чистяков, І. С. Ісламова, 2010

Використання *OpenMP* для розпаралелювання програм

Завдання паралельної програми полягає в тому, щоб організувати одночасне (паралельне) виконання незалежних операцій, функцій або блоків програмного коду. Реалізацію цього процесу можна навести на прикладі побудови літака. При отриманні замовлення на виробництво літака розробляються технічне завдання і креслення, за якими і буде здійснюватися його побудова. Але для створення силового агрегату не потрібно чекати, поки побудується планер, бортове устаткування або шасі, тобто окремі етапи виробництва можна організувати незалежно один від одного. А коли незалежні етапи будуть виконані, літак складається в цілому. Така організація роботи дає можливість істотно скоротити час виробництва літака.

Розпаралелювання програм виконується з метою більш ефективних затрат комп'ютерного часу на їх виконання. У паралельній програмі головний потік, з якого починається виконання будь-якої програми, зустрічаючи ключове слово, створює незалежні потоки, в яких йде виконання деяких операцій над абсолютно незалежними і не зв'язаними між собою даними. Якщо дані, які використовуються, будуть загальними для кількох потоків, що одночасно виконуються, то це призведе до помилок, які пов'язані не з правильністю роботи логіки програми, а з помилками обчислення або операцій. Це пов'язано з тим, що кожний потік може вносити зміни в загальні дані, аналогічно як функція змінює глобальну змінну. Спеціально для недопущення таких помилок існують спеціальні параметри розділу, за допомогою яких створюється змінна-копія для кожного потоку (аналогічно як копія глобальної змінної для використання всередині функцій). Після завершення виконання паралельних потоків управління програми передається знову головному потоку, який або продовжує виконання програми, або завершує її.

Схему роботи паралельної програми подано на рис. 1.

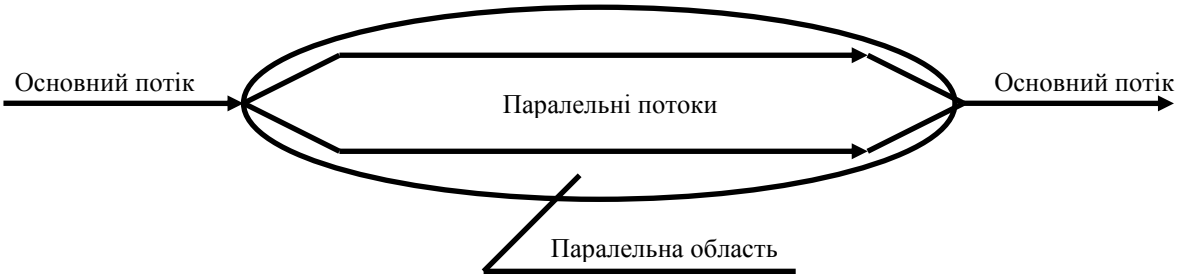

Рис. 1. Схема роботи паралельної програми

Для організації розпаралелювання програм з метою підвищення ефективності їх комп'ютерної реалізації на багатоядерних процесорах з загальною пам'яттю можна використовувати програмний засіб *OpenMP* (*Open Multi-Processing*) [2; 3]. Стандарт *OpenMP* було розроблено в 1997 р. як *API*, орієнтований на написання багатопотокових програмних додатків. Спочатку він був заснований на мові *Fortran*, але пізніше включив і Сі / С++. Остання версія *OpenMP* — 2.0; її повністю підтримує *Visual* C++ 2005.

Стандарт *OpenMP* підтримується також платформою Xbox \360.

OpenMP — це набір директив компілятора, бібліотечних процедур і змінних оточення, які призначені для програмування багатопотокових додатків на багатопроцесорних системах з єдиною пам'яттю на мовах Сі, С++ і *Fortran* [4].

Спочатку малося на увазі, що програма з *OpenMP* виконуватиметься на комп'ютері з декількома процесорами, причому кількість процесорів може не збігатися з кількістю потоків, на які розпаралелюється програма. Якщо кількість потоків більше кількості ядер або процесорів, то потоки можуть перемикатися при виконанні або «чекати» поки звільниться процесор. Якщо ж при ініціалізації багатопотоковій системі забороняється змінювати кількість потоків, то потоки виконуватимуться по черзі. Засоби, описані стандартом *OpenMP*, дають змогу маніпулювати потоками: видаляти, блокувати, перенаправляти, робити критичні секції, зупиняти і припиняти потоки і т. д. На однопроцесорних комп'ютерах паралельна програма виконуватиметься одним потоком.

Для подання послідовної програми у паралельному вигляді здебільшого використовують методи «інкрементного» програмування. Суть його полягає в тому, що програміст, проглядаючи програму, спочатку відшукує блоки, які можуть бути розпаралелені, а потім за допомогою спеціальних засобів *OpenMP* «розпаралелює» їх, тим самим зменшуючи кількість лінійного коду, і підвищуючи продуктивність програмного продукту або системи в цілому. Використання підходу «інкрементного» програмування полегшує

модернізацію раніше написаного програмного забезпечення.

Для реалізації паралельного виконання блоків програми потрібно додати в її код заголовний файл < omp.h >, необхідні директиви pragma і, якщо потрібно, скористатися відпо-відними функціями OpenMP періоду виконання. Директиви pragma — це директиви компілятора, які вказують як повинен оброблятися код, що слідує за pragma.

Для того, аби продемонструвати застосування *OpenMP* в одопотокових Сі/С++-програмах для розпаралелювання, наведемо його деякі основні директиви.

OpenMP підтримує директиви parallel, for, parallel for, section, sections, single, master, critical, flush, ordered, atomic, які визначають або механізми розподілу роботи (команд), або конструкції синхронізації.

Директиви pragma мають такий формат:

```
#pragma omp <директива> [розділ] [ [,]розділ] ]
```

Одна з найважливіших директив – parallel. Вона створює паралельний регіон для наступного за нею структурованого блоку, наприклад:

```
#pragma omp <parallel> [розділ] [ [,]розділ] ]
< структурований блок>
```

Ця директива повідомляє компілятор, що структурований блок коду повинен бути виконаний паралельно, в декількох потоках. Кожний потік виконуватиме один і той же потік команд, але не один і той же набір команд – все залежить від операторів, що управляють логікою програми, таких як if-else.

Директива #pragma omp for застосовується не для паралельного виконання коду, а для логічного розподілу групи потоків, щоб реалізувати вказані конструкції управляючої логіки. Вона повідомляє, що при виконанні циклу for в паралельному регіоні ітерації циклу повинні бути розподілені між потоками групи. Проте компілятор не розпізнає незалежність виконання циклів.

У наведеному нижче прикладі ітерації циклу не мають залежності, тобто одна ітерація циклу не залежить від результатів виконання інших ітерацій. Компілятор може автоматично розпаралелити цикли на декілька потоків.

```
#pragma omp parallel
{
#pragma omp for
for(int i =1; i < n; ++i)
      x[i] = y[i-1] + y[i+1];
}
```

Проте розпаралелювання циклу:

```
for(int i =1; i < n; ++i)
      x[i] = x[i-1] + y[i];
```

призведе до помилки, тому що для виконання *i*-ї ітерації необхідно знати результати *i*-1-ї ітерації, тобто ітерації циклу мають залежність.

Один з найважливіших атрибутів паралельного виконання програми — баланс навантаження (розподіл навантаження порівну між потоками). Баланс навантаження має велике значення, оскільки гарантує одночасну роботу всіх ядер. Без балансу навантаження деякі потоки можуть завершувати роботу значно раніше решти, що призводить до простою обчислювальних ресурсів і втрати продуктивності.

У циклах відсутність балансу навантаження є наслідком розходження часу обчислень у різних ітераціях циклу. У більшості випадків ітерації циклу займають однаковий час. В інших випадках можна знайти групи ітерації, що виконуються за однаковий час. Наприклад, іноді набір всіх парних ітерацій займає приблизно стільки ж часу, як і набір всіх непарних ітерацій. З іншого боку, може виявитися неможливим знайти набори ітерацій, що мають однаковий час виконання.

Отже, для балансування навантаження ядер комп'ютера доцільно надати *OpenMP* додаткову інформацію про планування циклу, аби він міг правильно розподілити ітерації циклу між потоками для оптимізації розподілу навантаження.

За умовчанням *OpenMP* припускає, що всі ітерації циклу займають однаковий час, і розподіляє ітерації циклу між потоками приблизно порівну.

Таким чином мінімізується ймовірність виникнення конфліктів пам'яті внаслідок її неправильного спільного використання.

Тому при розподілі циклів на дві великі частини (наприклад, на першу і другу половини) при використанні двох потоків імовірність накладення пам'яті виявляється меншою.

Слід зазначити, що в кінці паралельного регіону виконується бар'єрна синхронізація (barrier synchronization). Інакше кажучи, досягнувши кінця регіону, всі потоки блокуються до тих пір, поки останній потік не завершить свою роботу.

В програмах, які використовують OpenMP, змінні діляться на глобальні (shared), існуючі в одному екземплярі і доступні всім потокам, і локальні (private), існуючі в конкретному процесі.

Усі змінні середовища оточення і функції, що належать до *OpenMP*, повинні починатися з префікса `omp_`. Наприклад, функція для встановлення кількості потоків:

`omp_set_num_threads`, функція для визначення номера потоку:

`omp_get_threads_num`.

За допомогою цих функцій можна написати, наприклад, такий фрагмент програми:

```
        #pragma
omp_set_num_threads(3);
        if (omp_get_threads_num =
   = 2)
        <виконати індивидуальний код
для потоку за номером 2>
        else
        <код для всіх останніх пото-
ків >
```

Розглянемо на прикладі перемноження двох матриць $C = A \times B$ застосовування в програмі засобів *OpenMP* для її розпаралелювання.

Як відомо, перемноження двох матриць здійснюється за формулою:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} .$$

З цієї формули ми бачимо, що при перемноженні матриць обчислення проміжних добутків є незалежними, оскільки кожне з них записує (і читає) свій елемент c_{ij}. Таким чином, їх можна виконувати паралельно. Нижче наводиться фрагмент тексту програми на C++ з деякими вставками з *OpenMP*.

```
   #include <omp.h>
   …
   void mult_matrix (float
**mas1,float **mas2, long int x1,
                  long int y1, long
int y2)
   {
       float  **mas3;
       long int i,j,k;
       mas3=(float
**)malloc(sizeof(float*)*y2);
       for(i=0; i<y2; i++)
           mas3[i]=
(float*)malloc(sizeof(float)*x1);
   // заборонити змінювати кількість
потоків під час виконання
   #pragma omp_set_dynamic(0);
   // встановити кількість потоків,
що дорівнює 2
   #pragma omp_set_num_threads(2);
       for(i = 0; i < x1; i++)
           for(j = 0; j < y2; j++)
               mas3[i][j]= 0;
   // описуємо масиви, що містять
елементи матриць, глобально
видимими,
   // а індекси – локально видимими
   #pragma omp parallel shared(mas1,
mas2, mas3) private(i,j,k)
     #pragma omp for
       for (i = 0; i < x1; i++)
       {
         for (j = 0; j < y2; j++)
         { float s =0;
           for (k = 0; k < y1; k++)
           s += mas1[i][k] *
mas2[k][j];
         mas3[i][j]=s;
         }
       }
   free(mas3);
   }
```

Для підключення *OpenMP* до компілятора C++ потрібно виконати такі дії для налаштування проекту (рис. 2):

1. Зайти в середовищі розробки *Visual Studio* 2005 меню <Project> (Проект) та вибрати підпункт меню <name_project Properties> (назва_проекту Властивості) або скористуватися комбінацією клавіш Alt+F7.

2. У діалоговому вікні <name_project Property Pages> необхідно вибрати пункти меню <Configuration Properties> –> <C/C++> –> <Language> та навпроти опції <OpenMP Support> вибрати <Yes>.

Після виконання цих налаштувань компілятор починає сприймати команди та ключові слова *OpenMP* як директиви для виконання. Якщо не налаштовувати компілятор, то директиви сприйматися не будуть і програма буде виконуватись лінійно,а не паралельно. Співвідношення часу роботи програми без розпаралелювання до часу роботи програми з розпаралелюванням, тобто із застосуванням *OpenMP*, наведено в табл. 1.

Програма виконувалася на двоядерному процесорі. Конфігурація системи: DualCoe E6300 з 3.49ГБ оперативної пам’яті, операційна система – Windows XP. Час розв’язування подано в мікросекундах.

З табл. 1 видно, що час розв’язування задачі за паралельною програмою приблизно в два рази зменшується порівняно з часом розв’язування за звичайною програмою. Зі збільшенням кількості ядер ці показники будуть поліпшуватися. Отже, використання багатоядерності персонального комп’ютера є доцільним при розв’язуванні задач великих розмірів.

Суть другого експерименту полягала в тому, щоб розглянути, як виконуватиметься подана вище програма на одноядерному персональному комп’ютері (табл. 2). Розмір матриць 3000×3000.

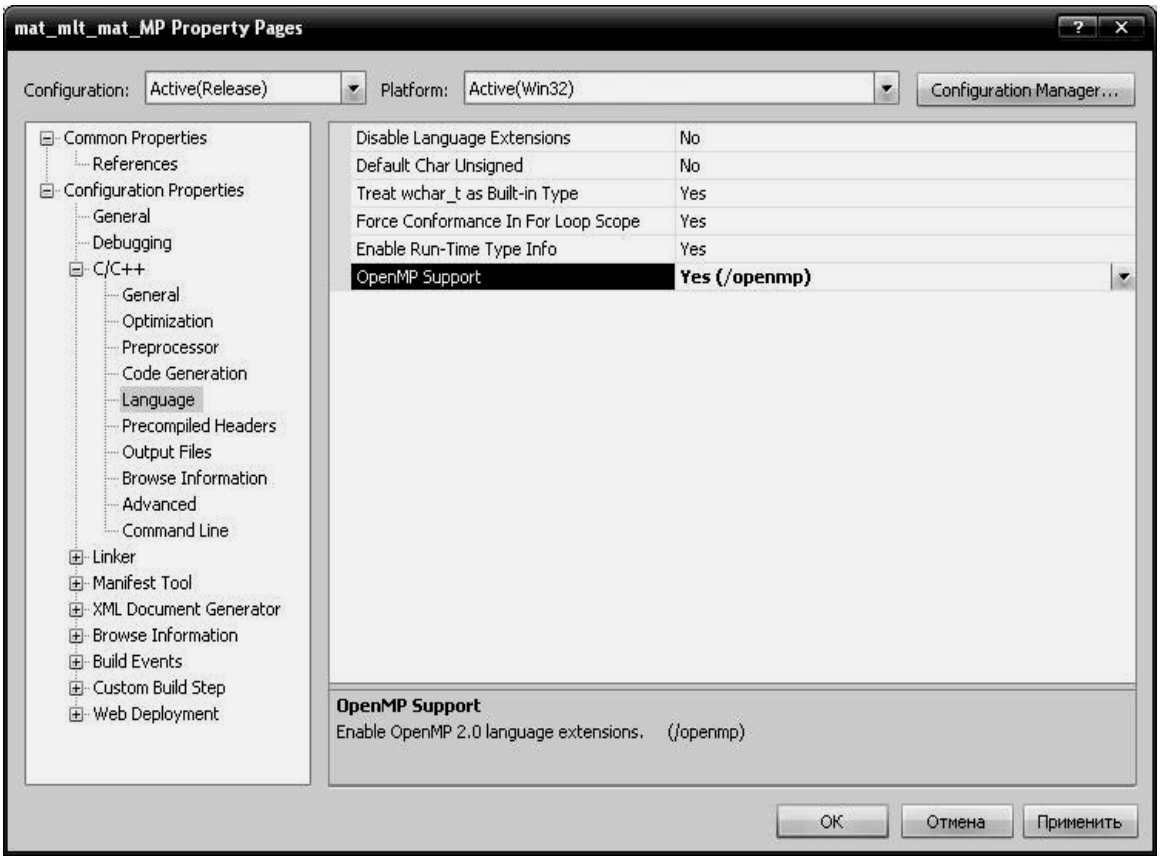

Рис. 2. Налаштування проекту на *OpenMP*

Таблиця 1

Співвідношення швидкості роботи різних варіантів програм

Розмір матриць	Програма з *OpenMP*	Програма без *OpenMP*
200×200	32	63
500×500	890	1764
1000×1000	8766	15703
2000×2000	71469	124469
3000×3000	264328	425687

Таблиця 2

Співвідношення швидкості роботи програм на різних комп'ютерах

Комп'ютер	Програма з *OpenMP*	Програма *без OpenMP*
Pentium4 3,0GHz +HT 2 ГБ ОЗУ	748156	730857
E6300 2x2,8GHz 3,49ГБ ОЗУ	243386	444954

Результат другого експерименту показує, що на *Pentium4* (одноядерний процесор з реалізацією технології HT) час розв'язування задачі за програмою із застосуванням *OpenMP* несуттєво відрізняється від часу розв'язування за звичайною програмою, оскільки тут програмно емулюються два ядра-потоки, а фізично комп'ютер має одне ядро.

Для використання повною мірою цієї технології(+15—20 % продуктивності) потрібно використовувати спеціальні API та функції, опис яких можна знайти на сайтах *Intel* і *Microsoft.*

Висновки

Проведені дослідження показують наскільки реальною і важливою стає можливість ефективного використання багатоядерності персонального комп'ютера для розв'язування задач великих

об'ємів. Час розв'язування задачі на багатопроцесорних системах з загальною пам'яттю, в тому числі на багатоядерних персональних комп'ютерах, значно скорочується при автоматичному розпаралелюванні програм за допомогою інтерфейсу *OpenMP*, що реалізує паралельне виконання як циклів, так і функціональних блоків програмного коду.

OpenMP підтримується практично всіма виробниками великих обчислювальних систем, легко інтегрується в системне програмне забезпечення багатоядерного комп'ютера. Технологія використання *OpenMP* дає змогу користувачам працювати з єдиним текстом для послідовної та паралельної програми, виконавши незначні вставки в програму. Крім того, оскільки *OpenMP* ґрунтується на «інкрементному» програмуванні, то користувачу немає необхідності одразу писати паралельну програму цілком, він її може створювати послідовно. Це значно спрощує процес створення паралельної програми.

У нових версіях компіляторів *Intel*, починаючи з версії 9.1, реалізовано розширення *OpenMP*, призначене для розпаралелювання програм для обчислювальних систем с розподіленою пам'яттю. Це розширення відоме під назвою *Cluster OpenMP*. Розв'язування задач із застосуванням *OpenMP* на персональних комп'ютерах не тільки підвищує ефективність їх використання, але також дає можливість краще зрозуміти суть паралельного програмування оволодіти навиками створення паралельних методів та програм розв'язування задач із застосуванням *Cluster OpenMP* на паралельних комп'ютерів, наприклад, кластерного типу.

ЛІТЕРАТУРА

1. *Воеводин В. В.* Параллельные вычисления / В. В. Воеводин, Вл. В. Воеводин. — С.Пб. : БХВ-Петербург, 2002. — С. 608.

2. *Антонов А. С.* Параллельное программирование с использованием технологии OpenMP / А. С. Антонов. — МГУ, 2009. — С. 78.

3. *Левин М. П.* Параллельное программирование с использованием OpenMP: учеб. пособие. — М. : Інтернет университет информационных технологий; БИНОМ. Лаборатория знаний, 2008. — С. 118.

4. *Voss M. J.* OpenMp shared memory parallel programming / M. J. Voss. — Toronto, Kanada, 2003. — P. 270.

5. *Parallel* programming in OpenMP. Morgan Kaufmann / R. Chandra, L. Dagum, D. Kohn, D. Maydan, J. McDonald, R. Menon. 2000. — P. 353.

Стаття надійшла до редакції 06.04.10.