УДК 004.272

Н.Г. Аксак, В.Д. Волонцевич, І.А. Осотов, Я.М. Філін
Харківський національний університет радіоелектроніки,
кафедра електронних обчислювальних машин

ДОСЛІДЖЕННЯ ЗБІЛЬШЕННЯ ПОТУЖНОСТІ У ПРОГРАМУВАННІ НА МУЛЬТИЯДЕРНІЙ КЛАСТЕРНІЙ СИСТЕМІ

Досліджено мультиядерну кластерну систему з використанням технології паралельного програмування. Проаналізовано паралельну та послідовну реалізації програми. Задля коректної та якісної програмної підтримки реалізації мультиядерної кластерної системи було використано технологію OpenMP та стандарт MPI.

Для розподілу обчислювального процесу за вузлами кластера було використано графічний інтерфейс користувача guiMPIRun версії 1.2.4.

In this work the exploring of multicore cluster system is considered. Different combinations of described technologies was applied. Parallel and consistent realizations of exploring program were examined. OpenMP and MPI standard's technologies for proper program support of multicore cluster system realization was applied.

Graphic user interface guiMPIRun version 1.2.4 for distribution of computing processes into cluster's nodes was used.

Вступ. Провідні виробники процесорів, такі як AMD та Intel, говорять про перспективний шлях розроблення комп'ютерів, який полягає в побудові комп'ютерних систем на основі мультиядерних процесорів. Сьогодні вже доступні процесори Intel Pentium D, Intel Celeron D, Intel Core 2 Duo, Intel Core 2 Quad**,** Intel Pentium Extreme Edition, AMD Athlon 64 X2, які мають дво- та чотири- ядерну архітектуру. В деяких моделях, на зразок Intel Pentium D та Intel Core 2 Duo, досягнуто повної фізичної двоядерної архітектури. Моделі, подібні до Intel Core 2 Duo Extreme Edition, містять два фізичні ядра, в кожному з яких по два логічних. Будування мультипроцесорних систем є першим та найголовнішим напрямком у мультипотоковості, але для цього шляху розроблення потрібні великі кошти. Використання мультиядерних систем є кращою можливістю для збільшення потужності процесорів.

Стандарт OpenMP був сформульований у 1997 як API задля написання портативних, багатопотокових рішень. Його було зроблено як стандарт на основі ФОРТРАН, але пізніше його доробили та ввели до нього підтримку C та C++ компіляторів. Поточна версія OpenMP 2.5 підтримує ФОРТРАН, C та C++. Компілятори Intel C++ та ФОРТРАН підтримують стандарт OpenMP версії 2.5. Програмна модель OpenMP забезпечує не залежний від платформи набір псевдокоментарів, директив, функціональних запитів, які явно інструктують компілятор, як і де використовувати паралелізм для розв'язання різних задач. Більшість циклів можна розпаралелити додаванням лише одного псевдокоментаря перед циклом. Завдяки тому, що рутинну роботу бере на себе компілятор та бібліотека періоду виконання OpenMP, більше уваги приділяється на з'ясування того, які цикли повинні бути розпаралелені та як краще змінити структуру алгоритмів задля збільшення потужності на багатоядерних процесорах.

Дослідження були проведені на кластерній системі на основі двоядерних комп'ютерів з процесорами Intel Pentium D. Було використано стандарт OpenMP задля розпаралелення програм та бібліотека MPI для розподілення робочих завдань вузлами кластера.

Основна частина. Обчислення коефіцієнта зростання потужності проводилось на програмі знаходження числа π (див. рис. 1) тому, що вона містить цикл однорідних ітерацій (послідовне складання часткових сум)

$$\pi = \frac{1}{n}\sum_{i=1}^{n}\frac{4}{1+x_i^2}; \quad x_i = \frac{1}{n}.$$

Якщо кількість ітерацій збільшиться, різниця у часі роботи послідовної та паралельної реалізації швидко зросте, тому що час чистих обчислень буде значно більшим, ніж накладні витрати на організацію паралельних обчислень, пересилку даних між вузлами кластера та їх синхронізацію. Мультипотоковість стає непотрібною, якщо використовується мала кількість подібних ітерацій.

```
#pragma omp parallel for private(x) shared(h) reduction(+:sum)
for (i = 1; i <= n; i ++)
{
x = h * ((double)i - 0.5);
sum += f(x);
}

pi = h * sum;
```

Рис. 1. Фрагмент програми (основна частина)

Програма, яка реалізує обчислення числа π, розкладається на функції для введення паралелізму. Використовуючи цей підхід, можна класифікувати конкретні задачі. Якщо дві з них можуть виконуватися одночасно, то вони запускаються розробником. Виконання задач паралельно у такий спосіб зазвичай потребує деяких модифікацій конкретних функцій для запобігання конфліктів та для демонстрації того, що ці задачі вже не потребують послідовного виконання.

У термінах програмування вдалим прикладом декомпозиції задач є текстовий редактор на зразок Microsoft Word. Коли користувач відкриває дуже великий документ, він одразу може починати вводити текст. Коли користувач вводить текст, процес нумерації сторінок проходить у фоновому режимі, про що можна пересвідчитися за швидким зростанням кількості сторінок у рядку стану. Введення тексту та нумерація сторінок – дві різні задачі, які розподілили програмісти на функції для паралельного виконання.

Розпаралелення циклу полягає у тому, що незалежні ітерації циклу виділяються у потоки та запускаються паралельно. В деякому розумінні, це трансформація переупорядкування, за якою оригінальна послідовність ітерацій циклу може бути переведена у невизначений порядок. Крім того, оскільки тіло циклу не є нероздільною операцією, оператори у двох різних ітераціях можуть виконуватися одночасно.

Теоретично достатньо перевести послідовний цикл у мультипотоковий, якщо у циклі немає залежностей. Тому перша проблема полягає як у визначенні, так і у зміні структури складних циклів, щоб впевнитися, що немає циклічних залежностей перед використанням псевдокоментарів OpenMP. Основний цикл, який реалізує обчислення числа π, складається з незалежних ітерацій. Ці ітерації дуже просто розділити на декілька блоків, що виконуються паралельно. Ця задача виконується псевдокоментарями стандарту OpenMP.

Псевдокоментарі ділять головний цикл на паралельні блоки автоматично та передають кожний з них відповідному ядру процесора.

Кожний експеримент проводився багато разів для досягнення стабільних значень, тому що була можливість нерівномірної загрузки вузлів кластеру малою кількістю ітерацій.

Отримані значення часу обчислень наведені в табл. 1, 2.

Таблиця 1

Обчислення з використанням мультиядерних переваг

Кількість вузлів	1	2	3	4	5
Кількість ітерацій	Час обчислень, с				
10000000	0,33	0,18	0,12	0,09	0,05
20000000	0,65	0,36	0,24	0,18	0,10
30000000	0,97	0,54	0,36	0,27	0,16
40000000	1,30	0,72	0,48	0,36	0,21
100000000	3,27	1,80	1,20	0,90	0,52
1000000000	32,79	17,94	12,00	9,02	5,21
2000000000	65,09	35,88	24,00	18,25	10,41
2147483640	70,02	38,66	25,76	19,36	12,03

Таблиця 2

Обчислення без використання мультиядерних переваг

Кількість вузлів	1	2	3	4	5
Кількість ітерацій	Час обчислень, с				
10000000	0,63	0,35	0,23	0,17	0,14
20000000	1,26	0,69	0,46	0,35	0,28
30000000	1,89	1,03	0,69	0,52	0,42
40000000	2,51	1,38	0,92	0,69	0,55
100000000	6,29	3,45	2,30	1,73	1,38
1000000000	62,86	34,60	23,14	17,30	13,84
2000000000	125,71	69,24	46,36	34,60	27,66
2147483640	134,93	74,27	49,64	37,15	29,71

Графічну форму отриманих результатів наведено на рис. 2, 3.

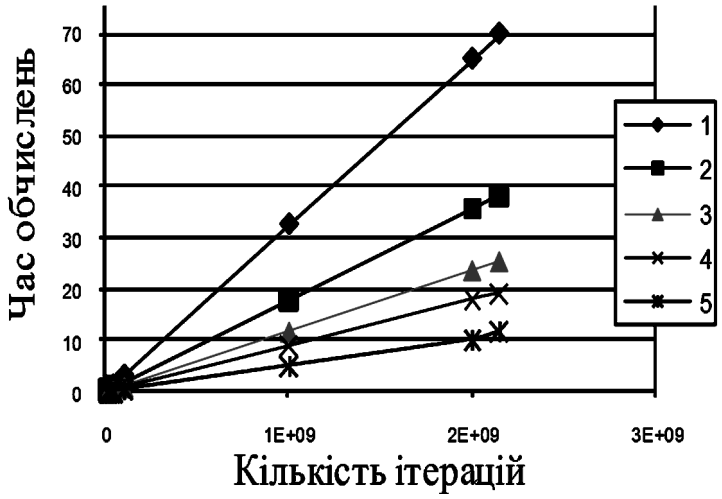

Рис. 2. Графік результатів обчислення з використанням мультиядерніх переваг

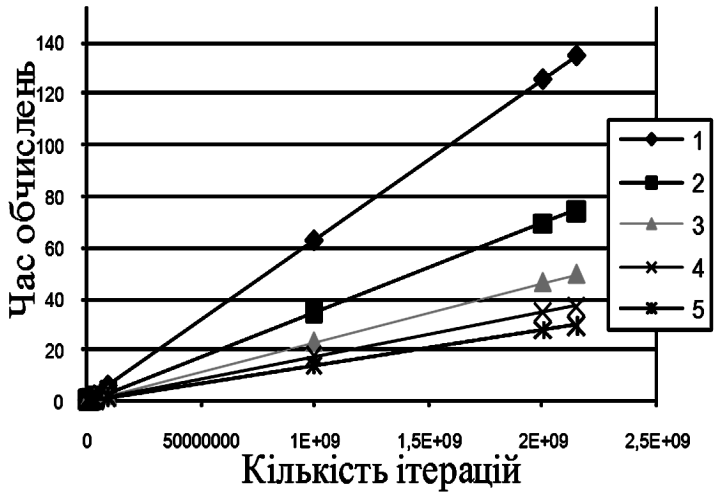

Рис. 3. Графік результатів обчислення без використанням мультиядерніх переваг

Аналіз результатів. Експериментально було визначено такі залежності:

- зростання потужності від відсотка навантаження паралельних ітерацій циклу;
- зростання потужності від відсотка паралельного коду в програмі;
- зростання потужності від кількості тотожних операцій, які виконуються паралельно;
- зростання потужності від кількості вузлів кластера;
- зростання потужності від кількості ядер процесора вузла.

Будь-яка програма має частини коду, що виконуються тільки послідовно (введення/виведення даних, визначення глобальних змінних), тому збільшення залежностей, зазначених вище, призводить до наближення коефіцієнта збільшення потужності до розміру, що дорівнює 100×C×N відсотків, де C – кількість ядер, N – кількість вузлів кластера. Були підтверджені фактори, які мають вплив на потужність паралельних програм та потрібні для створення мультипотокових програм.

Висновки. Для ефективної роботи програми було взято до уваги специфічні особливості комп'ютерної архітектури системи – двоядерного комп'ютерного кластера та проаналізовано:

- циклічні залежності;
- умови перегонів даних та інших факторів, які впливають на коректність та ефективність функціонування програми;
- орієнтацію на необхідну архітектуру комп'ютерної системи.

Досягнуто максимального рівня навантаження на всі ядра процесора великою кількістю (10000) виконаних обчислень програми.

Було знайдено співвідношення кількості послідовно до кількості паралельно виконуваних операцій, що призвело до зростання потужності у 11,216 раза.

1. Немнюгин С.А., Стесик О.Л. Параллельное программирование для многопроцессорных вычислительных систем. – СПб.: БХВ-Петербург, 2002. – 400 с. 2. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2004. – 608 с. 3. Камерон Хьюз, Трейси Хьюз. Параллельное и распределенное программирование на C++.: Пер. с англ. – М.: Изд. дом "Вильямс", 2004. – 672 с.