УДК 004.42

Ступаков В. О., магістр, група ПІ-49м
Яремчук С. І., канд. фіз.-мат. наук, проф.
Житомирський державний технологічний університет

БІБЛІОТЕКА OPENMP ДЛЯ РОЗРОБКИ БАГАТОПОТОКОВИХ ПРОГРАМ

Паралельне програмування застосовується тоді, коли для послідовної програми необхідно зменшити час її виконання, або коли послідовна програма у зв'язку з великим об'ємом даних, перестає поміщатися в пам'ять одного комп'ютера. Напрямок розвитку в області високопродуктивних обчислень саме направлено на вирішення цих двох задач: створення потужних обчислювальних комплексів з великим об'ємом оперативної пам'яті з одного боку і розробка відповідного програмного забезпечення з іншого.

OpenMP (Open Multi-Processing) – це набір директив компілятора, бібліотечних процедур і змінних середовища, які призначені для програмування багатопотокових додатків на багатопроцесорних системах із загальною пам'яттю. Цей інтерфейс став однією з найбільш популярних технологій паралельного програмування. OpenMP успішно використовується як при програмуванні суперкомп'ютерних систем з великою кількістю процесів, так і в настільних користувацьких системах або, наприклад, в Xbox 360. Бібліотека часто застосовується в математичних обчисленнях, оскільки дозволяє швидко і легко розпаралелити програму. Але при цьому ідеологія OpenMP не дуже вдало підійде, наприклад, для розробки серверного програмного забезпечення.

В OpenMP використовується модель паралельного виконання «розгалуження-об'єднання». Програма OpenMP починається як єдиний потік виконання, який називається початковим потоком. Коли потік зустрічає паралельну конструкцію, він створює нову групу потоків, яка складається з цього потоку і деякої кількості додаткових потоків, і стає головним потоком в новій групі. Всі члени нової групи (включаючи головний) виконують код всередині паралельної конструкції, в кінці якої є неявний бар'єр. Після паралельної конструкції виконання користувацького коду продовжує лише головний потік. В паралельну область можуть бути вкладені інші паралельні області, в яких кожен потік початкової області стає основним для своєї групи потоків. Вкладені області можуть у свою чергу включати області глибшого рівня вкладеності. Кількість потоків в групі, які виконуються паралельно, можна контролювати.

При використанні OpenMP в програму додаються два види конструкцій: функції середовища OpenMP і спеціальні директиви #pragma.

Функції OpenMP мають скоріше допоміжний характер, оскільки реалізація паралельності здійснюється за рахунок використання директив. Але в ряді випадків вони досить корисні і навіть необхідні. Функції можна розділити на три категорії: функції середовища, функції блокування/синхронізації і функції роботи з таймерами. Все ці функції мають імена, які починаються з omp_, и визначені у заготовочному файлі omp.h.

Конструкція #pragma в мові Сі/Сі++ використовується для задання додаткових вказівок компілятору. За допомогою цих конструкцій можна вказати як відбуватиметься вирівнювання даних в структурах, заборонити видавати попередження тощо. Форма запису: #pragma директиви.

Використання ключової директиви «omp» вказує на те, що команди відносяться до OpenMP. Таким чином директиви #pragma для роботи з OpenMP мають наступний формат:

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

Директиви OpenMp, як і будь-які інші директиви pragma, ігноруються тими компіляторами, які не підтримують дану технологію. При цьому програма компілюється без помилок як послідовна. Ця особливість дозволяє створювати добре переносимий код на базі технології OpenMP. Код буде виконуватися як послідовний, але це краще, ніж створювати дві гілки коду чи розставляти багато #ifdef.

OpenMP підтримує директиви private, parallel, for, section, sections, single, master, critical, flush, ordered і atomic та ряд інших, які визначають механізми розподілення роботи або конструкції синхронізації.

Найпопулярніший спосіб розподілення задач в OpenMP – паралельний цикл. Він дозволяє задати опцію schedule, яка змінює алгоритм розподілення ітерацій між потоками. Всього підтримується три таких алгоритми:

– статичне планування – при використанні цієї опції ітерації циклу будуть порівну (приблизно) поділені між потоками;

– блочно-циклічне розподілення ітерацій – кожний потік отримує задане число ітерацій на початку циклу, потім (якщо залишилися ітерації) процедура розподілення продовжується; планування виконується один раз.

– динамічне планування – кожний потік отримує задане число ітерацій, виконує їх і запитує нову порцію. На відміну від статичного планування, виконується багаторазово під час виконання програми.