OpenMP — део 2

Рачунарски системи високих перформанси

Петар Трифуновић Вељко Петровић

Факултет техничких наука
Универзитет у Новом Саду

Рачунарске вежбе, Зимски семестар 2022/2023.

Конструкције за синхронизацију извршавања задатака

- `taskwait` — синхронизација само задатака истог нивоа

```
#pragma omp taskwait
```

- `taskgroup` — синхронизује и подзадатке

```
#pragma omp taskgroup
        strukturirani-block
```

- `depend` — `task` клаузула

```
depend(in | out | inout : <lista-promenljivih>)
```

Пример 9: Синхронизација зависних задатака — *taskwait*

```
int y, z;
#pragma omp parallel
{
    #pragma omp single
    {
        #pragma omp task
        {
            #pragma omp critical
            printf ("Task 1\n");

            #pragma omp task
            {
                #pragma omp critical
                printf ("Task 2\n");
            }
        }

        #pragma omp taskwait

        #pragma omp task
        {
            #pragma omp critical
            printf ("Task 3\n");
        }
    }
}
```

- Задатак 3 ће се сигурно извршити након задатака 1, јер *taskwait* обезбеђује синхронизацију између њих.
- Задатак 2 ће се можда извршити пре задатка 3, јер *taskwait* не синхронизује задатке у поднивоу.

Пример 9: Синхронизација зависних задатака — *taskgroup*

```
int y, z;
#pragma omp parallel
{
    #pragma omp single
    {
        #pragma omp taskgroup
        {
            #pragma omp task
            {
                #pragma omp critical
                printf ("Task 1\n");

                #pragma omp task
                {
                    #pragma omp critical
                    printf ("Task 2\n");
                }
            }
        }

        #pragma omp task
        {
            #pragma omp critical
            printf ("Task 3\n");
        }
    }
}
```

- Сви задаци из групе *taskgroup* морају окончати своје извршење пре него што се изврши наредни задатак.
- Овиме се гарантује да ће се и задатак 1 и задатак 2 извршити пре задатка 3.

Пример 9: Синхронизација зависних задатака — *depend*

- `depend` — `task` клаузула
- Зависност постоји између задатака који имају истог директног родитеља и чије листе зависних променљивих деле бар једну променљиву.
- Типови зависности:
  - *in* тип зависности — задатак чека на све претходно генерисане задатке са *in*, или *inout* типом зависности (ако деле бар једну зависну променљиву)
  - *in,* или *inout* тип зависности — задатак чека на све претходно генерисане задатке било ког типа зависности (ако деле бар једну зависну променљиву)

Пример 9: Синхронизација зависних задатака — depend

```
#pragma omp parallel
{
        int y, z;
        #pragma omp single
        {
                #pragma omp task depend(out:y)
                y = f(x)
                #pragma omp task depend(in:y)
                z = g(y)
        }
}
```

- Други задатак мора да чека на први, јер:
  - Задаци деле зависну променљиву *y*,
  - Оба задатка генерисана су од стране једног истог, основног задатка,
  - Други тип зависности је *in*, први тип је *out*.

Задатак 8: Множење матрице и вектора

- Имплементирати секвенцијални програм за множење неквадратне матрице и вектора у *C* програмском језику.
- Након што се уверите да програм даје очекиване резултате, имплементирати *OpenMP* паралелни алгоритам на основу секвенцијалног програма.
- **Напомене:**
  - Свака од димензија матрице треба да буде макар 1000. За потребе тестирања програма димензије матрице могу да буду и мање.
  - Мерити извршавања програма функцијом *omp_get_wtime()*.

Задатак 9: Множење матрица — **домаћи**

- Имплементирати секвенцијални програм за множење две неквадратне матрице у *C* програмском језику.
- Након што се уверите да програм даје очекиване резултате, имплементирати *OpenMP* паралелни алгоритам на основу секвенцијалног програма.
- **Напомене:**
  - Коректан секвенцијални програм тестирати на великим матрицама (око 1000 по димензији, модификовати зависно од карактеристика рачунара на којем радите задатак). За потребе тестирања, димензије матрица могу да буду и мање.
  - Мерити извршавања програма функцијом *omp_get_wtime()*.

Задатак 10: Акумулирање вредности чворова стабла

- Направити секвенцијалну имплементацију стабла у C програмском језику. Сваки чвор стабла садржи један разломљени број у једнострукој прецизности. Потребно је имплементирати минимални скуп функционалности (креирање стабла, сабирање вредности чворова стабла и уништавање стабла).
- Након што се уверите да програм даје очекиване резултате имплементирати OpenMP паралелни програм на основу секвенцијалног програма.
- **Напомене:**
  - Паралелни програм имплементирати коришћењем $task$ конструкције.
  - Мерити извршавање програма функцијом *omp_get_wtime()*.

Задатак 11: Транспоновање матрице

- Дата је секвенцијална имплементација транспоновања матрице (директоријум *zadaci/matrix_transpose*). Направити *OpenMP* паралелну верзију алгоритма.
- **Напомене:**
  - Мерити извршавање програма функцијом *omp_get_wtime()*.

Задатак 12: Једноставни генетски алгоритам

- Дата је секвенцијална имплементација једноставног генетског алгоритма имплементираног у C програмском језику (директоријум *zadaci/genetic_algorithm*). Покренути секвенцијални алгоритам над свим датим примерима према упутству у *README.md* датотеци и анализирати времена извршавања делова генетског алгоритма.
- Одредити критичне делове кода и паралелизовати их коришћењем *OpenMP*.
- **Напомене:**
  - Мерити извршавање програма функцијом *omp_get_wtime()*.

Задатак 13: Тражење корена функције над интервалом — домаћи

- Дата је секвенцијална имплементација методе за одређивање корена функције над задатим интервалом методом бисекције (директоријум *zadaci/bisection*). Покренути секвенцијални алгоритам над свим датим примерима према упутству у *README.md* датотеци и погледати решења сва три задата примера.
- Затим имплементирати *OpenMP* паралелно решење.
- **Напомене:**
  - Алгоритам испробати на још примера. При провери тачности добијеног решења могуће је користити неки од алата за одређивање корена функције (Волфрам, на пример).
  - Мерити извршавање програма функцијом *omp_get_wtime()*.

Коришћени материјали

- Документација са *OpenMP* сајта, видети www.openmp.org
  "Introduction to OpenMP", Tim Mattson, доступно на овом линку
  "Introduction to OpenMP", пратећа презентација
- "Parallel Computing Book", Victor Eijkhout, електронска верзија књиге је доступна на овом линку