Система с общей памятью OpenMP

Практическое занятие #2

OpenMP

Open Multi-Processing

Открытый стандарт для распараллеливания программ для языков C, C++, fortran.

Пример программы с OpenMP

```
#include<omp.h>
#include<stdio.h>

int main(int argc,char **argv){

    #pragma omp parallel
    printf(“Hello, World!\n”);

    return 0;
}
```

Результат работы

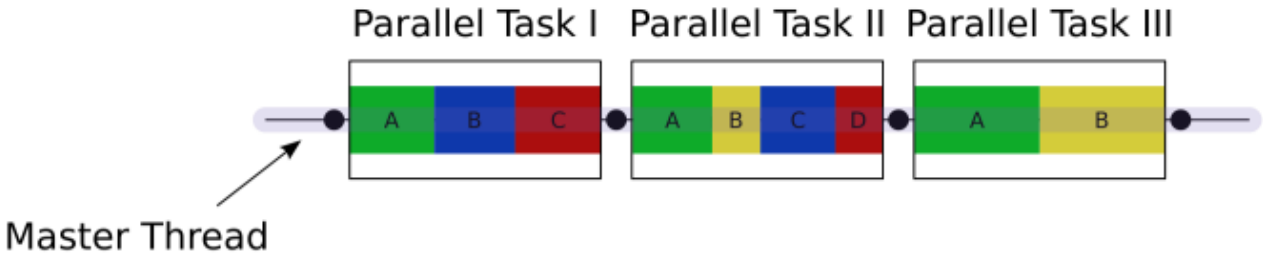
- ```
  g++ hello.cpp -o hello
  ./hello
  Hello, World!
  ```

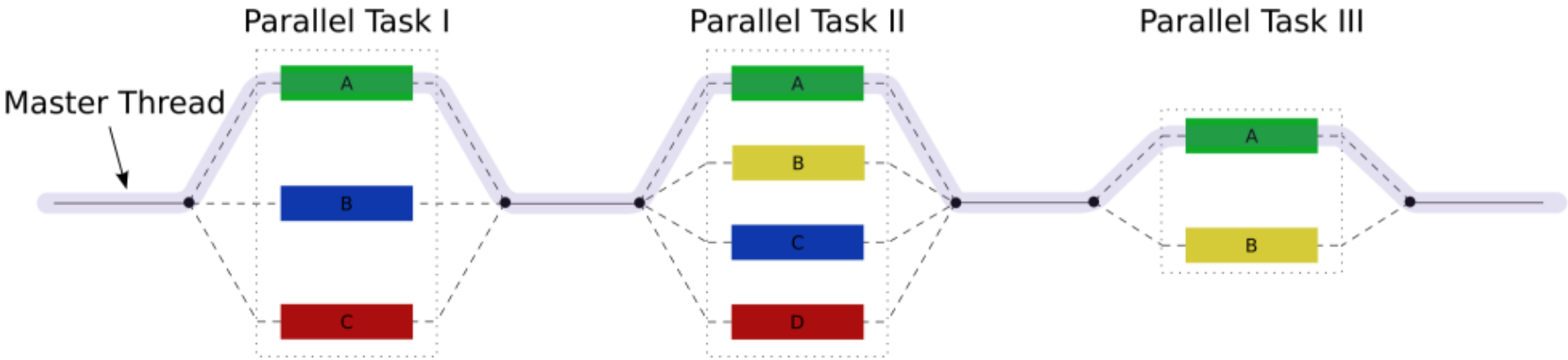
- ```
  g++ hello.cpp -fopenmp -o hello
  ./hello
  Hello, World!
  Hello, World!
  …
  ```

OpenMP

- Интерфейс параллельного программирования для систем с общей памятью
- Включает в себя набор директив компилятора, библиотечных функций и переменных окружения
- Разрабатывается с 1997 года

Принцип работы OpenMP





Синтаксис OpenMP

- Директивы
  `#pragma omp directive [clause[clause[…]]]`

- Функции
  `omp_function()`

- Переменные окружения
  `OMP_PARAM`

Количество потоков в OpenMP

- Директива
  `#pragma omp parallel … num_threads(N)`
- Функции
  `omp_set_num_threads(N)`
- Переменные окружения
  `set OMP_NUM_THREADS=N`

Основные элементы OpenMP

- Создание потоков
- Распределение работы по потокам
- Управление работой с данными
- Синхронизация потоков

Независимые потоки

```
#pragma omp parallel
{
     int myid = omp_get_thread_num();

     if (myid == 0) {
         do_something();
     } else {
         do_something_else(myid);
     }
}
```

Независимые потоки

```
#pragma omp parallel sections
{
    #pragma omp section
    {
        // Thread 1
    }
    #pragma omp section
    {
        // Thread 2
    }
     …
    #pragma omp section
    {
        // Thread K
    }
}
```

Параллельное выполнение цикла

```
#pragma omp parallel for private(i)
for(i = 0; i < K; i++) {
  …
}
```

```
int A[K], B[K];

#pragma omp parallel for private(i) shared(A,B)
for(i = 0; i < K; i++) {
    A[i] *= B[i];
}
```

Reduction

```
int A[K]
int sum = 0;

#pragma omp parallel for shared(A) reduction(+:sum)
for(i = 0; i < K; i++) {
    sum += A[i];
}
```

Вычисление числа Пи

$$\pi = \sum_{i=0}^{N} \frac{4}{1 + {x_i}^2} \qquad x_i = \frac{\left(i + \frac{1}{2}\right)}{N}$$

```
int Pi= 0;
double step = 1.0 / N;
double x, Pi = 0;

for(i = 0; i < N; i++) {
    x = step * (0.5 + i);
    Pi += 4.0 / (1 + x*x);
}
```

Вычисление числа Пи

```
int count = omp_get_max_threads()
double tmp[count];

#pragma omp parallel private(i,x) shared(tmp)
{
    int id = omp_get_thread_num();
    for(i = id; i < N; i += count) {
        x = step * (0.5 + i);
        tmp[id] += 4.0 / (1 + x*x);
    }
}

for(i = 0; i < count; i++) {
    Pi += tmp[i];
}
```

Вычисление числа Пи

$$\pi = \sum_{i=0}^{N} \frac{4}{1 + x_i^2} \qquad x_i = \frac{\left(i + \frac{1}{2}\right)}{N}$$

```
int Pi= 0;
double step = 1.0 / N;
double x, Pi = 0;

#pragma omp parallel for private(x,i) reduction(+:Pi)
for(i = 0; i < N; i++) {
    x = step * (0.5 + i);
    Pi += 4.0 / (1 + x*x);
}
```

ВОПРОСЫ?