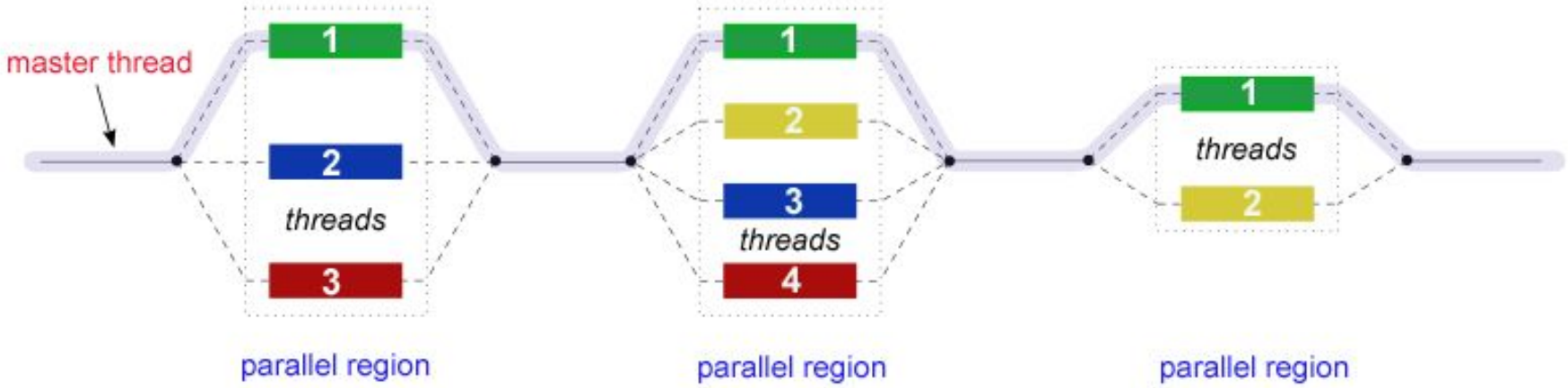
Средства и системы параллельного программирования

Семинар #6. OpenMP

Fork-join model



Директивы в OpenMP

```
#pragma omp directive-name [clause[ [,] clause] ... ] new-line
```

Виды директив:

- parallel Construct
- Loop-Related Directives
- Combined Constructs
- Worksharing Constructs
- Synchronization Constructs and Clauses

parallel Construct

```
#pragma omp parallel [clause[ [,] clause] ... ] new-line
   structured-block
```

Директива для создания параллельной области

полезные clause:

num_threads(integer-expression)
default(firstprivate, none, private, shared)
private(list)
firstprivate(list)
shared(list)
proc_bind(master | close | spread)

Число потоков в OpenMP-программе

- export OMP_NUM_THREADS=8 - указать в переменной окружения, что вы хотите запускать программу на 8 потоков (остаётся, пока вы продолжаете работу в сессии терминала)
- OMP_NUM_THREADS=8 ./a.out - установить переменную окружения только на время выполнения команды
- По умолчанию в программе будет запускаться столько потоков, сколько на процессоре ядер

- `void omp_set_num_threads(int num_threads);` - функция для вызова внутри кода, до параллельной области
- num_threads в clause у директивы parallel

omp_get_thread_num() - идентификация номера потока (от 0 до num_threads)

Single (worksharing construct)

#pragma omp single [clause[[,] clause] ...] new-line

structured-block

Директива, указывающая, что структурный блок должен выполниться только одним (произвольным) потоком

полезные clause:

private(list)

firstprivate(list)

copyprivate(list)

Master (worksharing construct)

#pragma omp master new-line

structured-block

То же самое, что и single, только явно обозначаем, что структурный блок должен выполняться потоком с номером 0

Критические секции

#pragma omp critical [(name) [[,] hint(hint-expression)]] new-line

structured-block

Директива, указывающая, что структурный блок может единовременно вызывать только один поток

Loop-Related Directives

Коллеги, на семинаре, естественно, мной была допущена ошибка - omp for позволяет внутри параллельной области раздать итерации по потокам. Итераций НЕ будет в *num_threads* раз больше, чем предусмотрено циклом.
(В отличие от далее упоминаемого taskloop)

#pragma omp for [clause[[,] clause] ...] new-line

for-loops

Директива, предписывающая разделить итерации цикла между потоками

полезные clause:
private(list)

firstprivate(list)

lastprivate([lastprivate-modifier:] list)

reduction([reduction-modifier,]reduction-identifier : list)

schedule([modifier [, modifier]:]kind[, chunk_size])

nowait

Combined parallel-loop directive

#pragma omp parallel for [clause[[,] clause] ...] new-line

for-loops

Директива, объединяющая parallel и for - удобно, когда нам нужно создать параллельные области только для выполнения цикла

В качестве clause можно использовать любые clause из parallel и for

Измерение времени

`double omp_get_wtime(void);` - потокобезопасная функция, ей можно измерять время выполнения одного потока

```
double start;
double end;
start = omp_get_wtime();
... work to be timed ...
end = omp_get_wtime();
```

Вложенный параллелизм

void omp_set_max_active_levels(int max_levels); - задание максимальной вложенности параллельных областей

void omp_set_dynamic(int dynamic_threads); - дать возможность уменьшать/увеличивать размер команды в каждом parallel при нехватке ресурсов

void omp_set_nested(int nested); - во многих реализациях устаревший способ задания возможности вложенного параллелизма

Sections (worksharing construct)

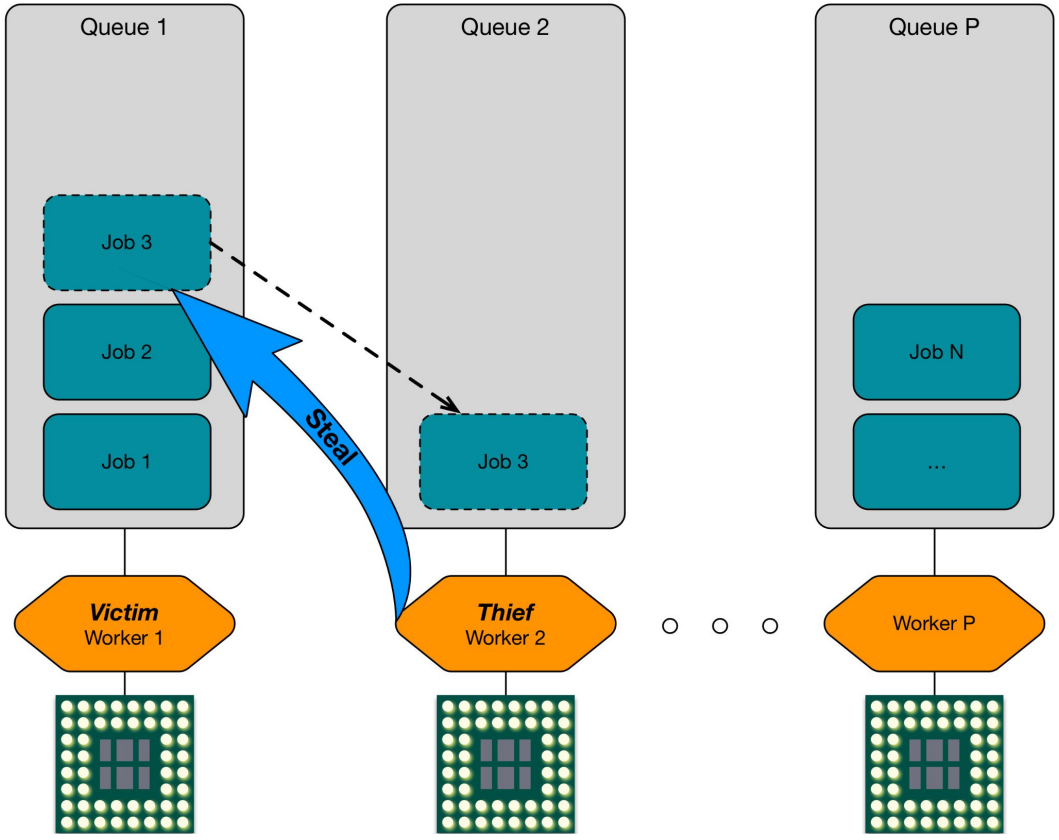
```
#pragma omp sections [clause[ [,] clause] ... ] new-line
   {
   [#pragma omp section new-line]
      structured-block
   [#pragma omp section new-line
      structured-block]
   ...
   }
```

Возможность создать отдельные блоки кода, выполняющимися потоками в параллельной области.

Каждая секция выполняется каким-либо одним потоком из параллельной области

Распределение секций между потоками в целом случайно.

Work stealing



Сложные случаи для распараллеливания

```
while ( <expr> ) {
    ...
}
```

```
void myfunc( <args> )
{
    ...
    myfunc( <newargs> );
    ...
}
```

Tasks

#pragma omp task [clause[[,] clause] ...] new-line

structured-block

Создание task. Task можно воспринимать как отчуждаемую часть задачи. Поток, создающий task, может выполнить код в блоке самостоятельно, а может передать выполнение кода другому потоку.

Полезные clause:

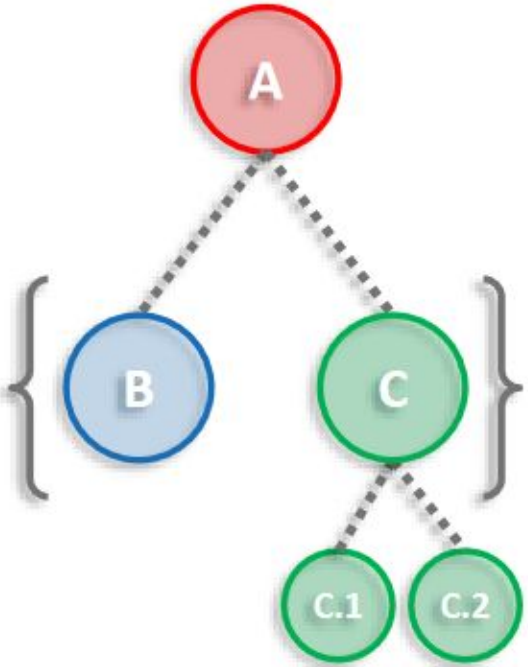
`if([ task :] scalar-expression)` - позволяет заставить поток тут же выполнить блок при scalar-expression, сводящемуся к false. Полезно, когда размер задачи уже достаточно мал

`private(list), firstprivate(list), shared(list)`

Task synchronization

```
#pragma omp taskwait
```

директива для выполнения ожидания всех дочерних task. Работает только для детей (первого поколения), не работает для task вложенности больше 1.



#pragma omp barrier

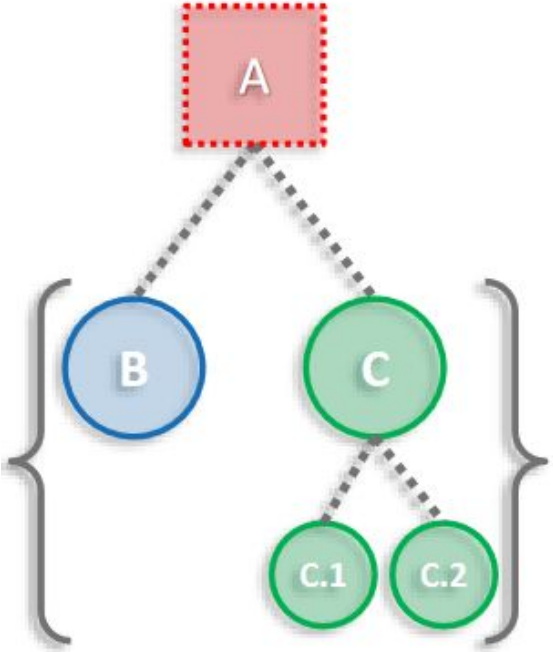
внутри parallel блока, если мы не находимся в work-sharing блоке, барьер может быть использован для ожидания выполнения всех task.

Task synchronization

#pragma omp taskgroup [clause[[,] clause]...]

{structured-block}

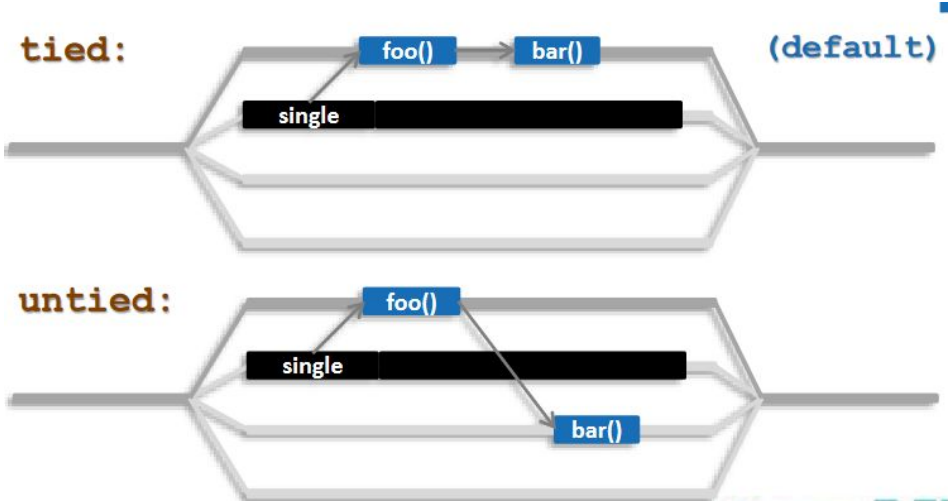
Директива taskgroup позволяет создать группу из task и контролировать (синхронизировать) выполнение всех task, порождённых в ней (в том числе сколь угодно вложенных)



Планирование внутри task

taskyield - директива, говорящая, что поток хочет уйти с ядра. Tied task (по умолчанию) призывает потом продолжить выполнение в этом же потоке, untied - снимает эту привязку. Работает не во всех реализациях.

```
#pragma omp parallel
#pragma omp single
{
    #pragma omp task untied
    {
    foo();
    #pragma omp taskyield
    bar();
    }
}
```



Taskloop

```
#pragma omp taskloop [clause[[,] clause] ...] new-line
    for-loops
```

Директива, позволяющая в цикле создавать наборы task, причём каждый такой набор образуется в полноценную taskgroup

Полезные clause:

`grainsize(grain-size)` - число итераций для каждого task

`num_tasks(num-tasks)` - число task, по которым нужно распределить итерации цикла

Задание

Используя OpenMP tasks (и, возможно, sections), реализовать решение задачи N-Queens:

Расставить N ферзей на шахматную доску N×N так, чтобы ни один ферзь не бил никакого другого (ни по горизонтали, ни по вертикали, ни по диагоналям). Требуется посчитать число решений.

В отчёте написать краткий комментарий относительно того, как реализовывали задание и график (либо таблицу) ускорения при использовании некоторого множества значений числа потоков.

Подсказка: возможно, наиболее удобно будет идти по строкам и создавать таски исходя из разных позиций для постановки в этой строке ферзя. Для учёта диагоналей, возможно, стоит использовать битовый вектор и сдвиги

Дедлайн: 3.11, 10.11