Advanced OpenMP: Tasks

Philipp Samfass
Scientific Computing in Computer Science / INF

Summer Term 2018

Advanced OpenMP : Tasks

- Brief overview
  - introduced with OpenMP 3.0
  - motivated by applications that are not naturally implemented with (parallel) loops such as recursive applications
  - programmer specifies **tasks** which are structured code blocks
  - OpenMP runtime manages task execution and scheduling
  - execution of tasks by OpenMP threads
  - synchronization constructs (e.g., taskwait or barrier) to ensure that tasks have been executed at some point
  - tasks may recursively create child tasks
  - tasks may be suspended temporarily and re-scheduled at a later point, potentially on a different thread
  - work-stealing of tasks between threads

Advanced OpenMP : Tasks

- compiler directives
  - task construct

```
#pragma omp task [clause, ···] newline
{
    structured block
}
```

    - creates an explicit task
    - needs to be inside a parallel region
    - encountering thread may or may not immediately execute the task: any thread in the current team may execute the task
    - data scoping constructs as for other OpenMP pragmas: default, private, firstprivate, shared
    - tasks are guaranteed to be finished at an implicit or explicit barrier or using the taskwait directive…

Advanced OpenMP : Tasks

- synchronization construct
  - taskwait construct

#pragma omp taskwait *newline*

  - encountering task is suspended until all child tasks of current task have finished (**important**: only direct children but not their children are completed!)

Advanced OpenMP : Tasks

- tasking
  - example 1

```
#pragma omp parallel default (shared)
{
  #pragma omp single
  {
    #pragma omp task
      printf ("hello world\n");
    #pragma omp task
      printf ("hello again\n");
  }
}
```

  - output?

Advanced OpenMP : Tasks

- tasking
  - example 2

```
#pragma omp parallel default (shared)
{
  int x=0;
  #pragma omp single
  {
    #pragma omp task
    {
      x++;
      printf ("from task 1: x = %d\n", x);
    }
    #pragma omp taskwait
    #pragma omp task
    {
      x++;
      printf ("from task 2: x = %d\n", x);
    }
  }
}
```

Advanced OpenMP : Tasks

- tasking
  - example 3: parallel computation of the nth Fibonacci number

```
int fib(int n)
{
  int i,j;
  if (n<2) return n;
  else {
    #pragma omp task firstprivate(n) shared(i)
    i=fib(n-1);
    #pragma omp task firstprivate(n) shared(j)
    j=fib(n-2);
    #pragma omp taskwait
    return i+j;
  }
}
```