



CS475
OpenMP Tasks

Wim Bohm, CS, CSU

source: Oracle OpenMP API User's Guide

Dynamic tasks in OpenMP

- 🌐 OpenMP specification version 3.0 introduced a new feature called **tasking**. Tasks are generated dynamically in recursive structures or **while** loops.
- 🌐 In OpenMP, an **explicit** task is specified using the task directive. The task directive defines the code associated with the task and its data environment. The task construct can be placed anywhere in the program; whenever a thread encounters a task construct, a new task is generated.

Task Execution

- 🌐 When a thread encounters a task construct, it may choose to execute the task immediately or defer its execution until a later time.
- 🌐 If task execution is deferred, then the task is placed in a pool of tasks.
- 🌐 A thread that executes a task may be different from the thread that originally encountered it.

Data environment 1

- The task directive takes the following data attribute clauses that define the data environment of the task:
 - default (private | firstprivate | shared | none)
 - private (*list*)
 - firstprivate (*list*)
 - shared (*list*)
- All references within a task to a variable listed in the shared clause refer to the variable with that same name known immediately prior to the task directive.
- For each private and firstprivate variable, new storage is created and all references to the original variable in the lexical extent of the task construct are replaced by references to the new storage. A firstprivate variable is initialized with the value of the original variable at the moment the task is encountered.

Data environment 2

- The task directive takes the following data attribute clauses that define the data environment of the task:
 - default (private | firstprivate | shared | none)
 - private (*list*)
 - firstprivate (*list*)
 - shared (*list*)
- The OpenMP 3.0 specification describes how the data-sharing attributes of variables referenced in parallel and task
- The OMP parallel construct creates “implicit” tasks

Data environment 3

- 🌐 The rules for how the default data-sharing attributes of variables are implicitly determined may not always be obvious. To avoid any surprises, it is recommended that the programmer explicitly scope all variables that are referenced in a task construct using the data sharing attribute clauses, rather than rely on the OpenMP implicit scoping rules.

Task Wait

- 🌐 **The TASKWAIT Directive**
- 🌐 The taskwait directive specifies a wait on the completion of children tasks generated since the beginning of the current (implicit or explicit) task.
- 🌐 The taskwait directive specifies a wait on the completion of direct children tasks, not all descendent tasks.

Example: nfib

// nfib counts the number of nodes in the fib call tree

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


nfib's main

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

```
...
```

```
#pragma omp parallel shared(n,v)
```

```
{
```

pragma omp parallel:
needed to create the parallel threads
executing the tasks

```
#pragma omp single
```

```
v=nfib(n);
```

pragma omp single:
one thread executes the initial nfib(n)
call

```
}
```

```
...
```

```
}
```

Number of tasks in nfib

🌐 #tasks nfib(n) = nfib(n)

nfib(30) = 2,692,537

WAY too many tasks created

tasks do nothing but tasks creation

We need to prune the task tree

Pruning the task tree 1

```
int nfib(long n) {  
    long i, j;  
    if (n<2) return 1;  
    else {  
        #pragma omp task shared(i) if (n>33)  
        i=nfib(n-1);  
        #pragma omp task shared(j) if (n>33)  
        j=nfib(n-2);  
        #pragma omp taskwait  
        return i+j+1;  
    }  
}
```

Two tasks get spawned and
the parent task does nothing!

Better if the parent task does
one of the nfibs

Pruning the task tree 2

```
int nfib(long n) {  
    long i, j;  
    if (n<2) return 1;  
    else {  
        #pragma omp task shared(i) if (n>33)  
            i=nfib(n-1);  
        j=nfib(n-2);  
        #pragma omp taskwait  
        return i+j+1;  
    }  
}
```