OpenMP Tutorial

Dirk Schmidl
IT Center, RWTH Aachen University
Member of the HPC Group
schmidl@itc.rwth-aachen.de

Christian Terboven
IT Center, RWTH Aachen University
Head of the HPC Group
terboven@itc.rwth-aachen.de

OpenMP Overview

IWOMP Tutorial: October 5th, 2016

Christian Terboven

Members of the OpenMP Language Committee

OpenMP Overview - Topics

- Core Concepts
- Synchronization

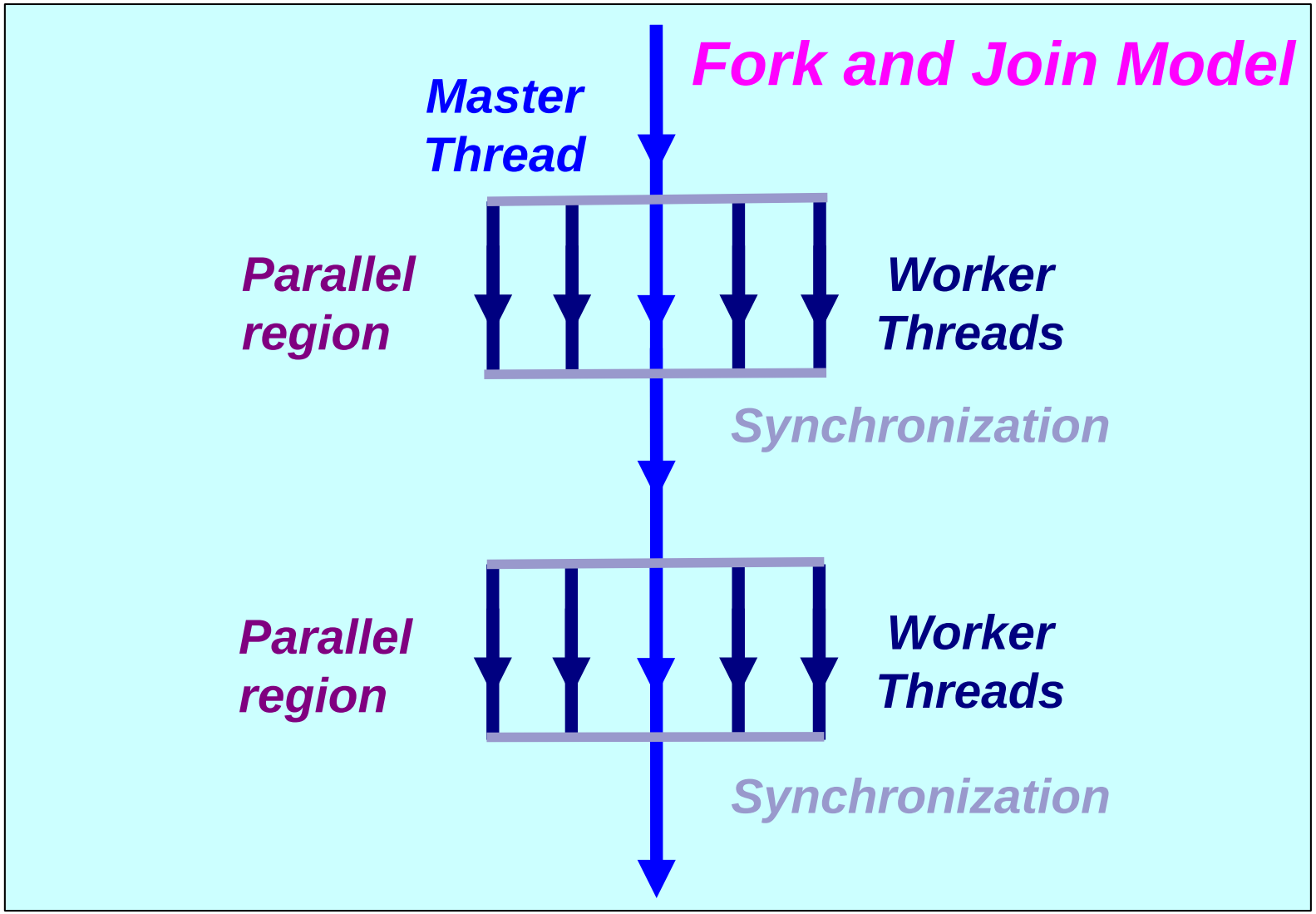
Defining Parallelism in OpenMP

A parallel region is a block of code executed by all threads in the team

```
#pragma omp parallel [clause[[,] clause] ...]
{
   "this code is executed in parallel"

} // End of parallel section (note: implied barrier)
```

```
!$omp parallel [clause[[,] clause] ...]

   "this code is executed in parallel"

!$omp end parallel (note: implied barrier)
```

The OpenMP Execution Model



The Worksharing Constructs

```
#pragma omp for
{
   ....
}

!$OMP DO
   ...
!$OMP END DO
```

```
#pragma omp sections
{
      ....
}

!$OMP SECTIONS
      ....
!$OMP END SECTIONS
```

```
#pragma omp single
{
      ....
}

!$OMP SINGLE
      ....
!$OMP END SINGLE
```

- ✓ *The work is distributed over the threads*
- ✓ *Must be enclosed in a parallel region*
- ✓ *Must be encountered by all threads in the team, or none at all*
- ✓ *No implied barrier on entry*
- ✓ *Implied barrier on exit (unless the nowait clause is specified)*
- ✓ *A work-sharing construct does not launch any new threads*

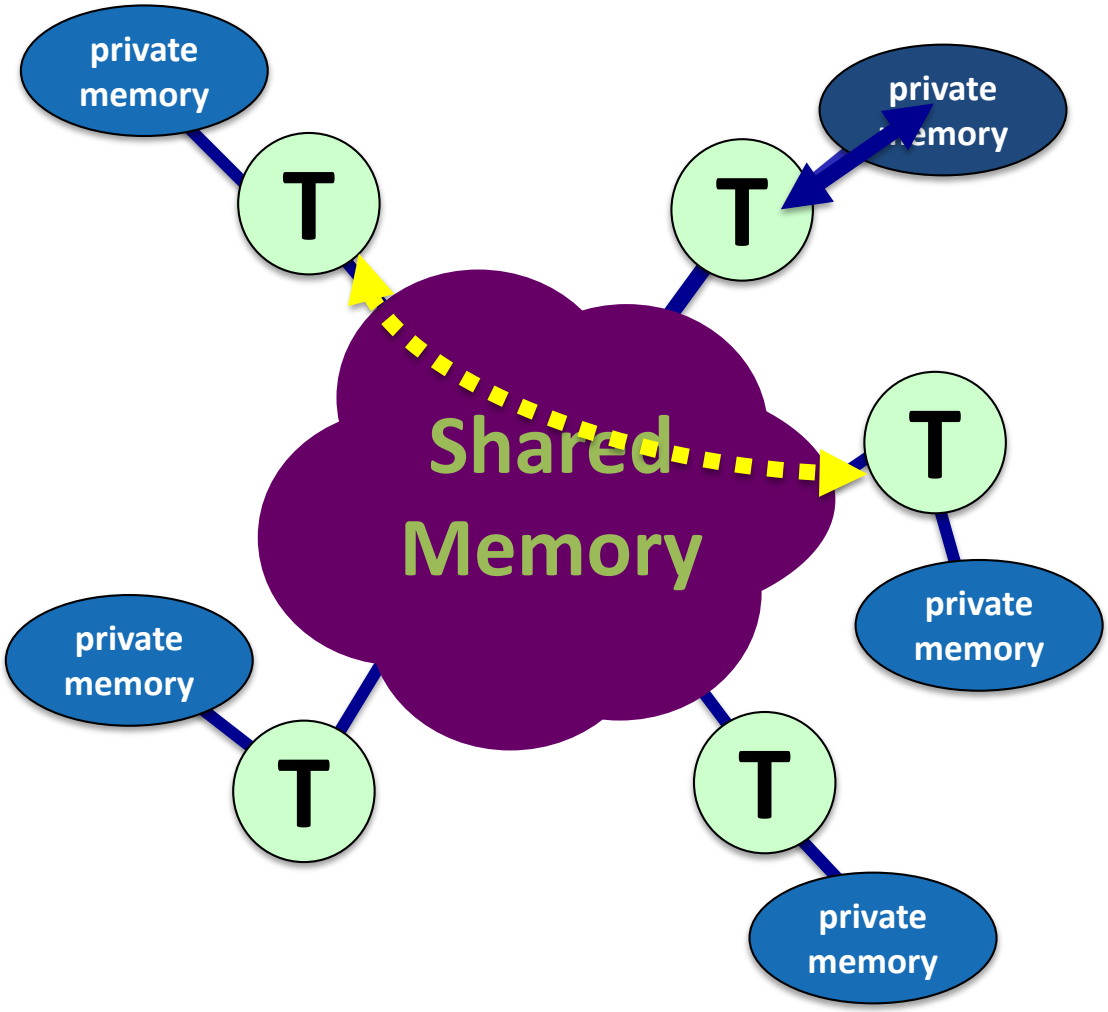
The Single Directive

Only one thread in the team executes the code enclosed

```
#pragma omp single [private][firstprivate] \
                   [copyprivate][nowait]
{
    <code-block>
}
```

```
!$omp single [private][firstprivate]
    <code-block>
!$omp end single [copyprivate][nowait]
```

The OpenMP Memory Model

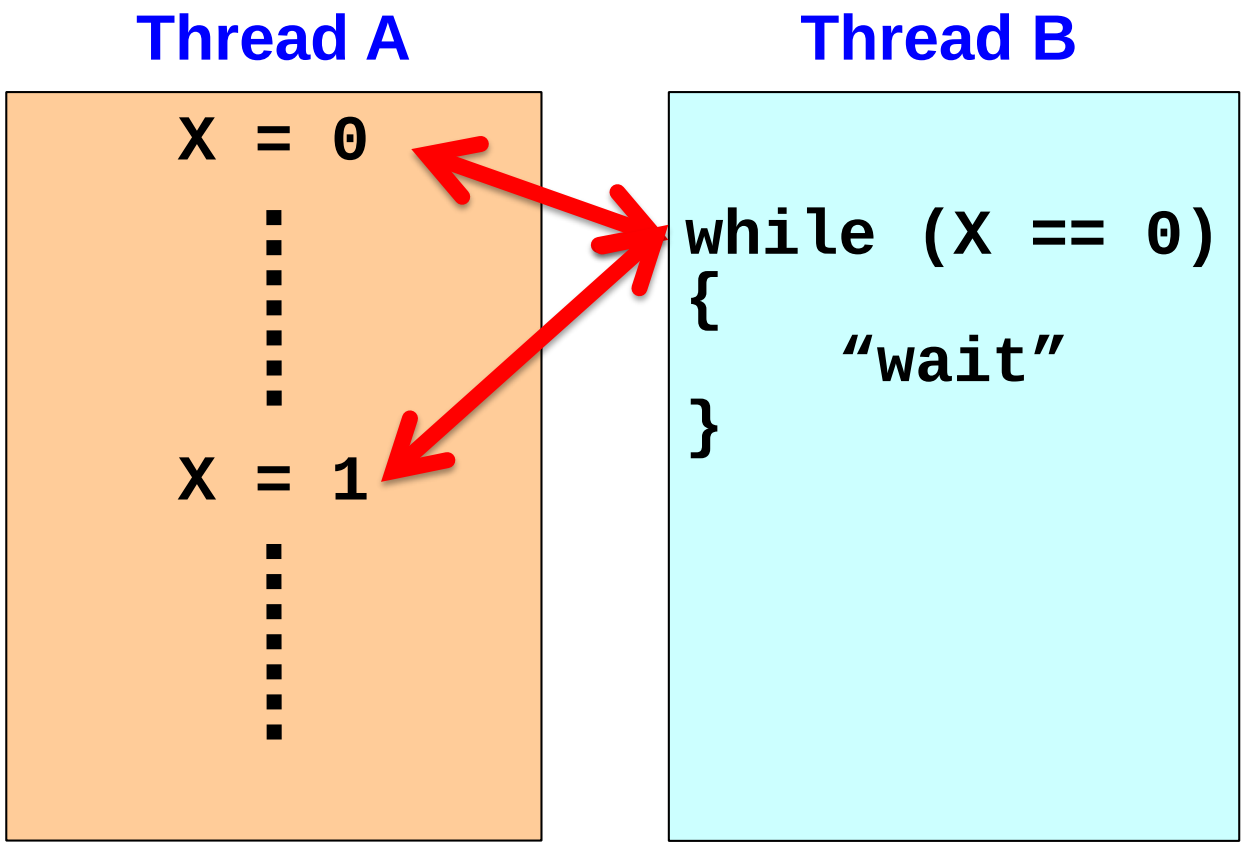


- *All threads have access to the same, globally shared memory*
- *Data in private memory is only accessible by the thread owning this memory*
- *No other thread sees the change(s) in private memory*
- *Data transfer is through shared memory and is 100% transparent to the application*

Gotcha's

- Need to get this right
  - → Part of the learning curve
- Private data is undefined on entry and exit
  - → Can use firstprivate and lastprivate to address this
- Each thread has its own temporary view on the data
  - → Applicable to shared data only
  - → Means different threads may temporarily not see the same value for the same variable ...
  - → Let me explain

The Flush Directive



If shared variable X is kept within a register, the modification may not be made visible to the other thread(s)

About The Flush

- Strongly recommended: do **not** use this directive
  - → … unless really necessary. Really ☺.
  - → Could give very subtle interactions with compilers
  - → If you insist on still doing so, be prepared to face the OpenMP language lawyers
- Implied on many constructs
  - → A good thing
  - → This is your safety net

The OpenMP Barrier

- Several constructs have an implied barrier
  - →This is another safety net (has implied flush by the way)
- In some cases, the implied barrier can be left out through the “nowait” clause
- This can help fine tuning the application
  - →But you’d better know what you’re doing
- The explicit barrier comes in quite handy then

```
#pragma omp barrier
```

```
!$omp barrier
```

The Nowait Clause

- To minimize synchronization, some directives support the optional nowait clause
  - → If present, threads do not synchronize/wait at the end of that particular construct
- In C, it is one of the clauses on the pragma
- In Fortran, it is appended at the closing part of the construct

```
#pragma omp for nowait
{
        :
}
```

```
!$omp do
        :
        :
!$omp end do nowait
```