

# Programming OpenMP

## *Parallel Region*

Christian Terboven



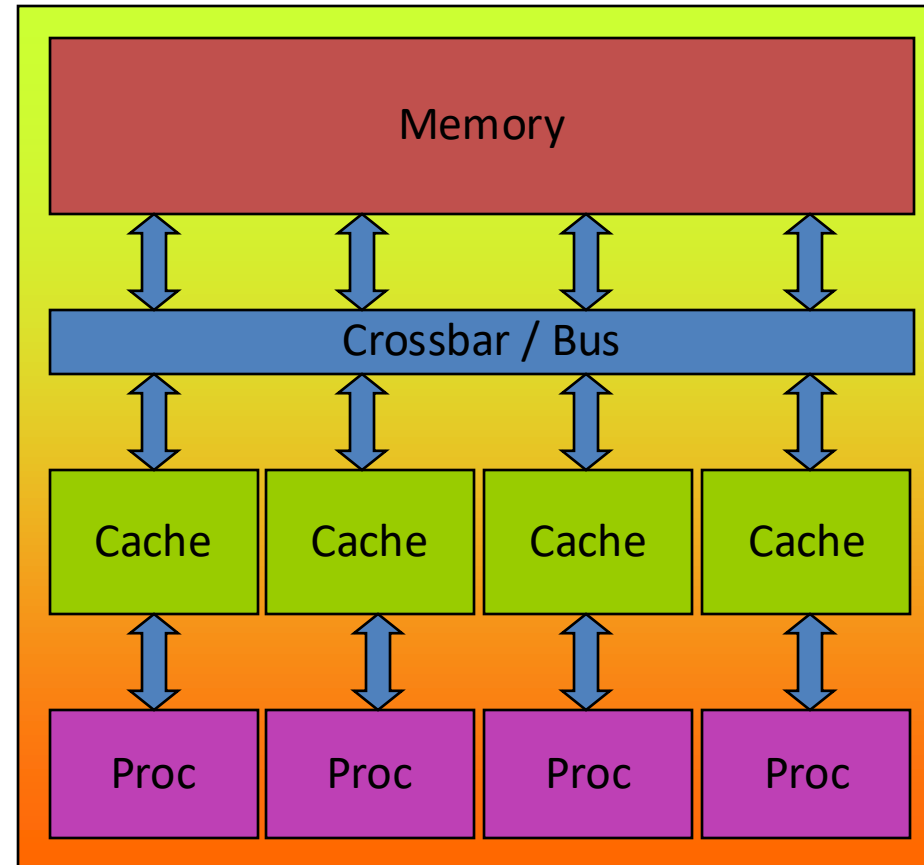
## OpenMP's machine model

- OpenMP: Shared-Memory Parallel Programming Model.

All processors/cores access a shared main memory.

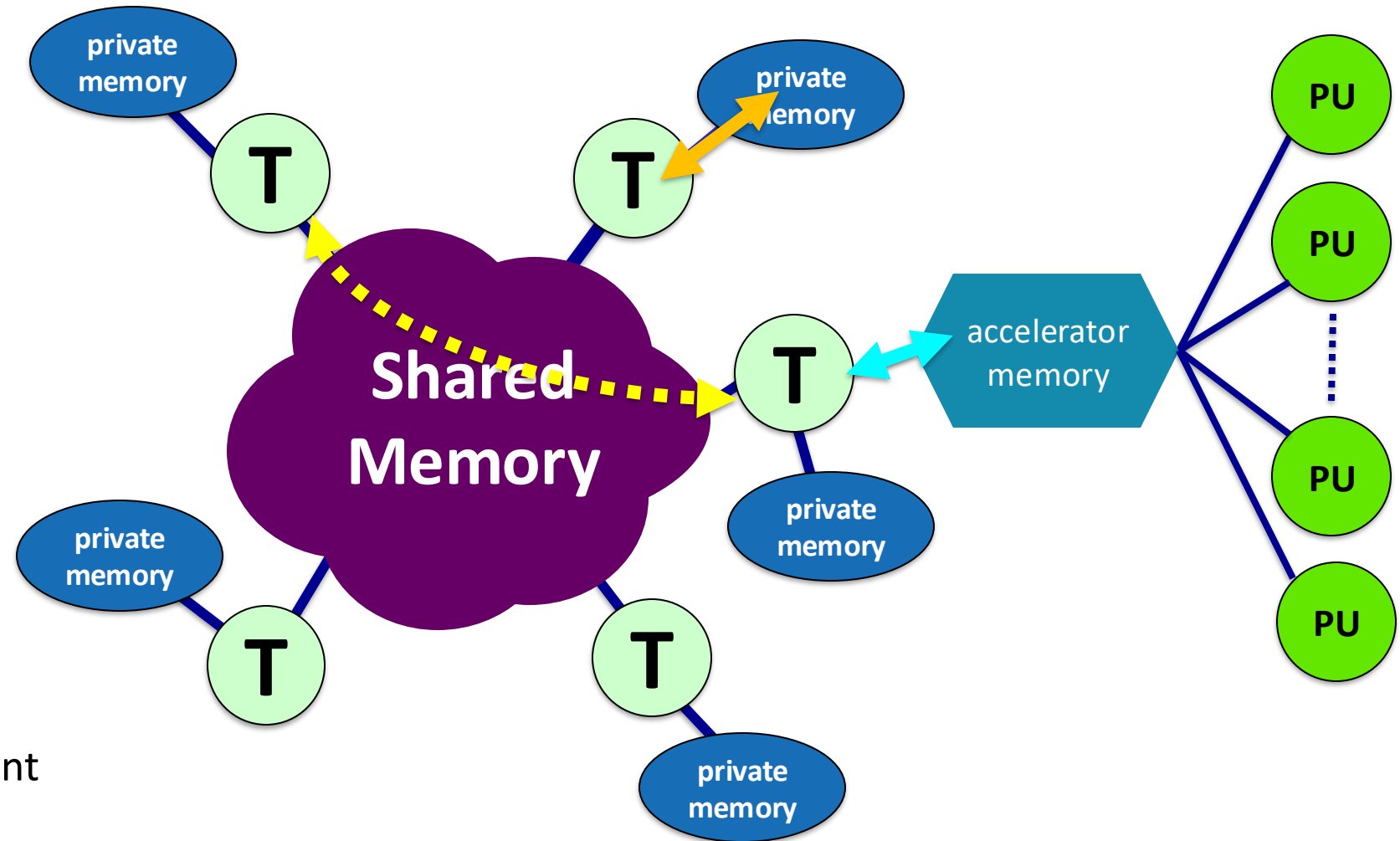
Real architectures are more complex, as we will see later / as we have seen.

Parallelization in OpenMP employs multiple threads.



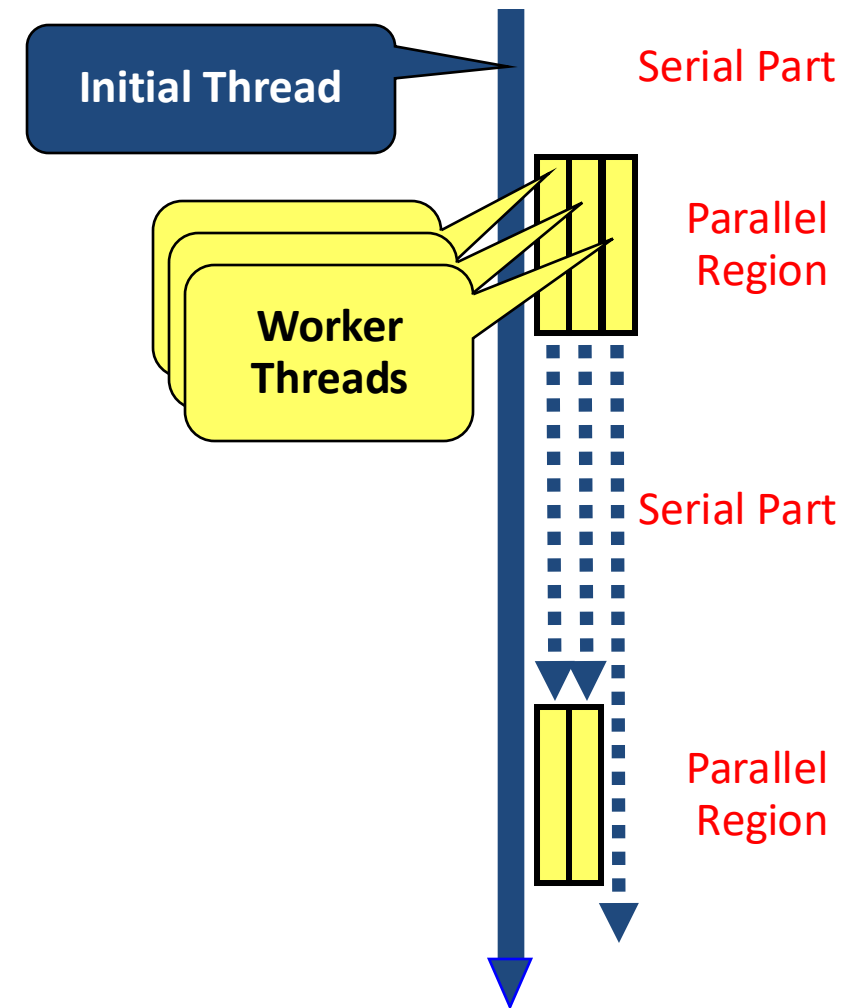
## The OpenMP Memory Model

- 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



## The OpenMP Execution Model

- OpenMP programs start with just one thread: The *Initial* thread.
- *Worker* threads are spawned at *Parallel Regions*, together with the Initial thread they form the *Team* of threads.
- In between *Parallel Regions* the *Worker* threads are put to sleep. The OpenMP *Runtime* takes care of all thread management work.
- Concept: *Fork-Join*.
- Allows for an incremental parallelization!



## Parallel Region and Structured Blocks

- The parallelism has to be expressed explicitly.

C/C++

```
#pragma omp parallel
{
    ...
    structured block
    ...
}
```

Fortran

```
!$omp parallel
    ...
    structured block
    ...
!$omp end parallel
```

- *Structured Block*

- Exactly one entry point at the top
- Exactly one exit point at the bottom
- Branching in or out is not allowed
- Terminating the program is allowed (abort / exit)

- *Specification of number of threads:*

- Environment variable: OMP\_NUM\_THREADS=...
- Or: Via `num_threads` clause:  
add `num_threads (num)` to the parallel construct

## Starting OpenMP Programs on Linux

- From within a shell, global setting of the number of threads:

```
export OMP_NUM_THREADS=4  
./program
```

- From within a shell, one-time setting of the number of threads:

```
OMP_NUM_THREADS=4 ./program
```

# Hello OpenMP World