HPC.NRW

OpenMP in Small Bites

Dr. Christian Terboven

Overview

Dr. Christian Terboven

OpenMP in Small Bites

History

HPC.NRW

- De-facto standard for Shared-Memory Parallelization.

- 1997: OpenMP 1.0 for FORTRAN
- 1998: OpenMP 1.0 for C and C++
- 1999: OpenMP 1.1 for FORTRAN
- 2000: OpenMP 2.0 for FORTRAN
- 2002: OpenMP 2.0 for C and C++
- 2005: OpenMP 2.5 now includes both programming languages.

- 05/2008: OpenMP 3.0
- 07/2011: OpenMP 3.1

- 07/2013: OpenMP 4.0

- 11/2015: OpenMP 4.5

- 11/2018: OpenMP 5.0

OpenMP™

http://www.OpenMP.org

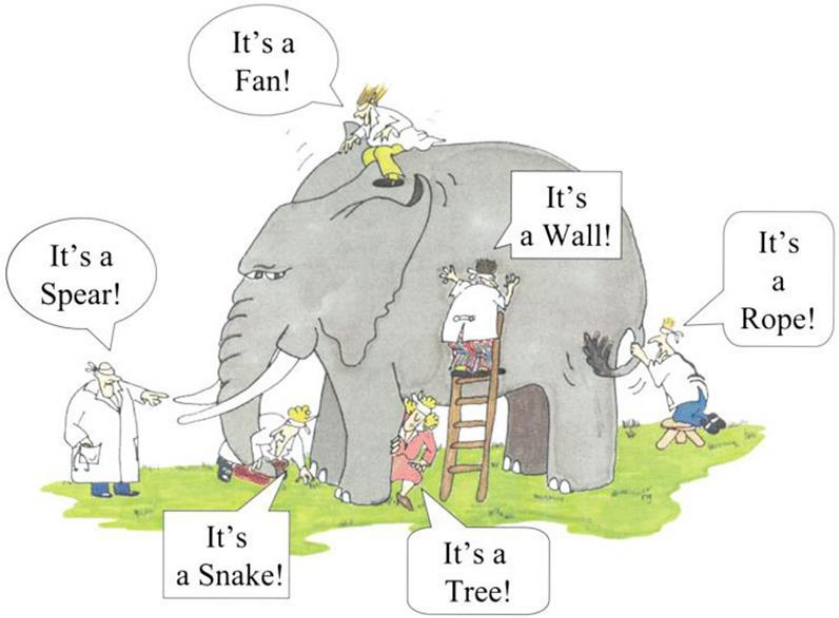
RWTH Aachen University is a member of the OpenMP Architecture Review Board (ARB) since 2006.

What is OpenMP?

HPC.NRW

- De-facto standard Application Programming Interface (API) to write shared memory parallel applications in C, C++, and Fortran

- Compiler Directives, Runtime routines and Environment variables



Introduction to OpenMP

Dr. Christian Terboven

OpenMP Overview & Parallel Region

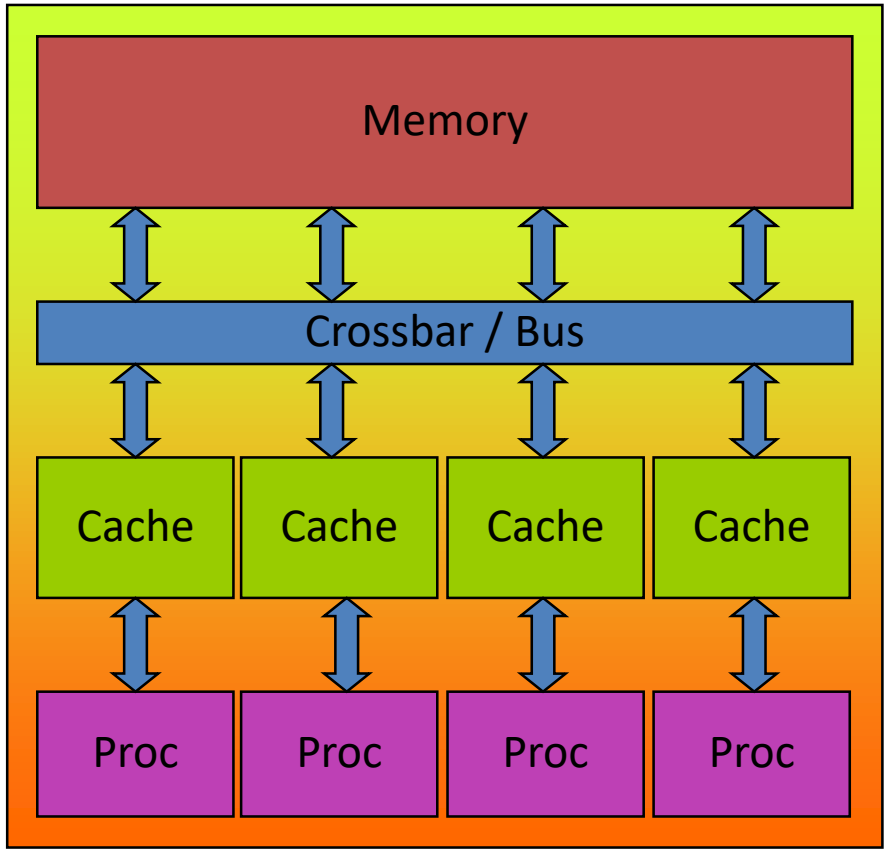
OpenMP‘s machine model

– 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.

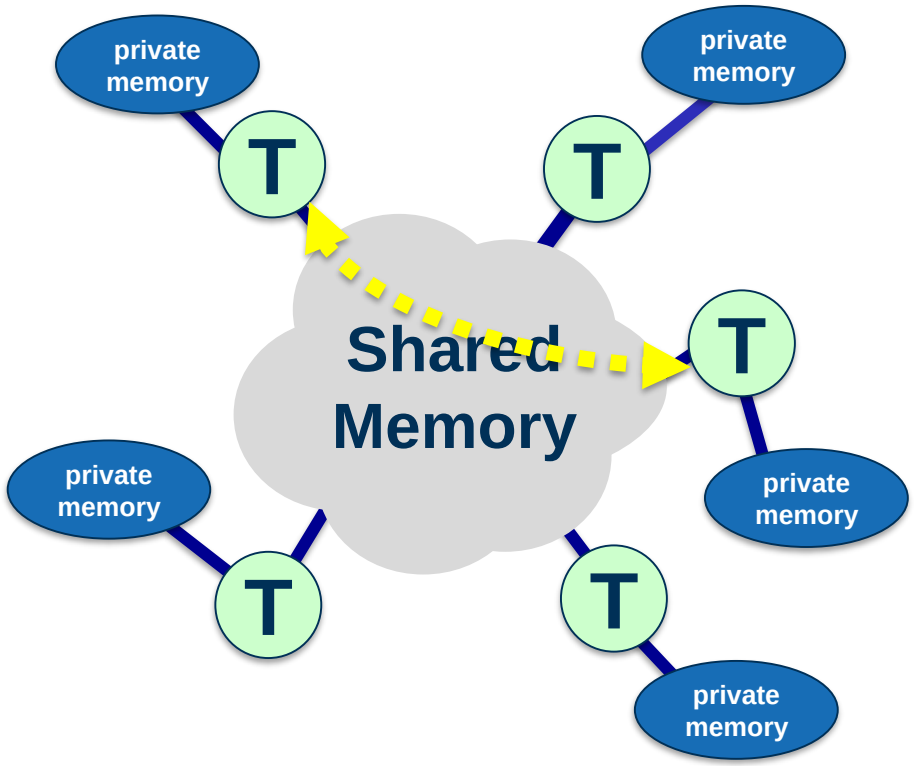
Parallelization in OpenMP employs multiple threads.



OpenMP Memory Model

HPC.NRW

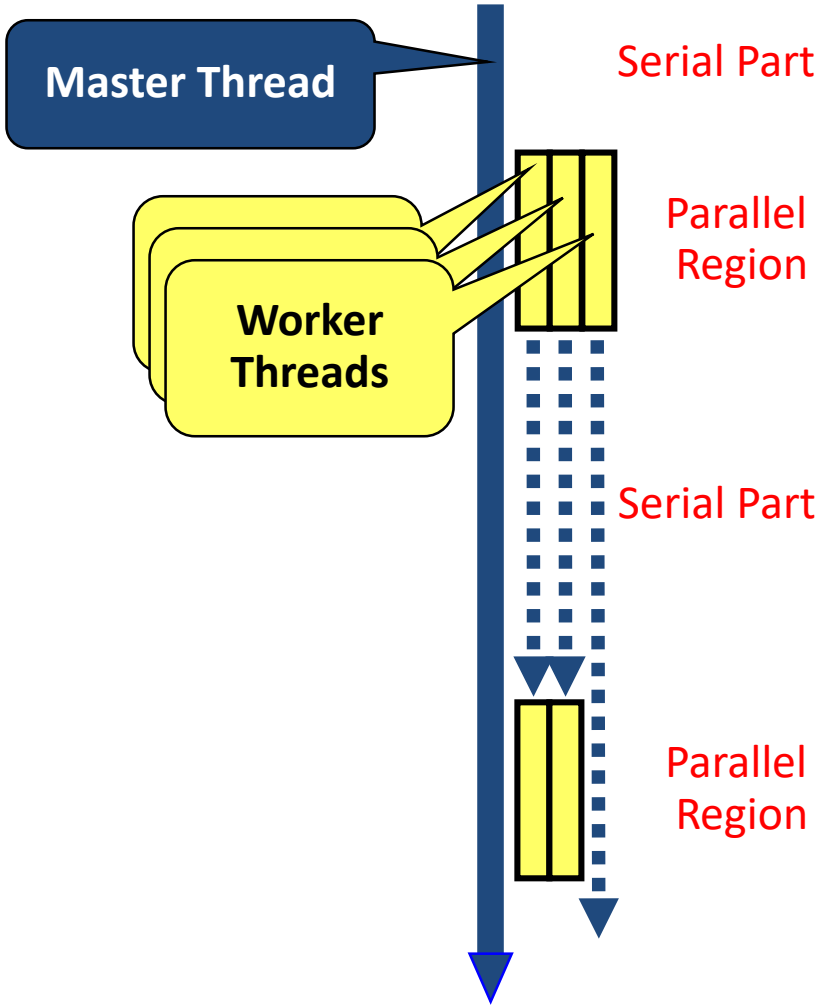
- All threads have access to the same, globally shared memory
- Data in private memory is only accessible by the thread owning this memory
- No other thread sees the change(s) in private memory
- Data transfer is through shared memory and is 100% transparent to the application



OpenMP Execution Model

HPC.NRW

- OpenMP programs start with just one thread: The *Master*.
- *Worker* threads are spawned at *Parallel Regions*, together with the Master 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

HPC.NRW

- 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

Introduction to OpenMP

Dr. Christian Terboven

Hello OpenMP World

Starting OpenMP Programs on Linux

HPC.NRW

– 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
```

Questions?