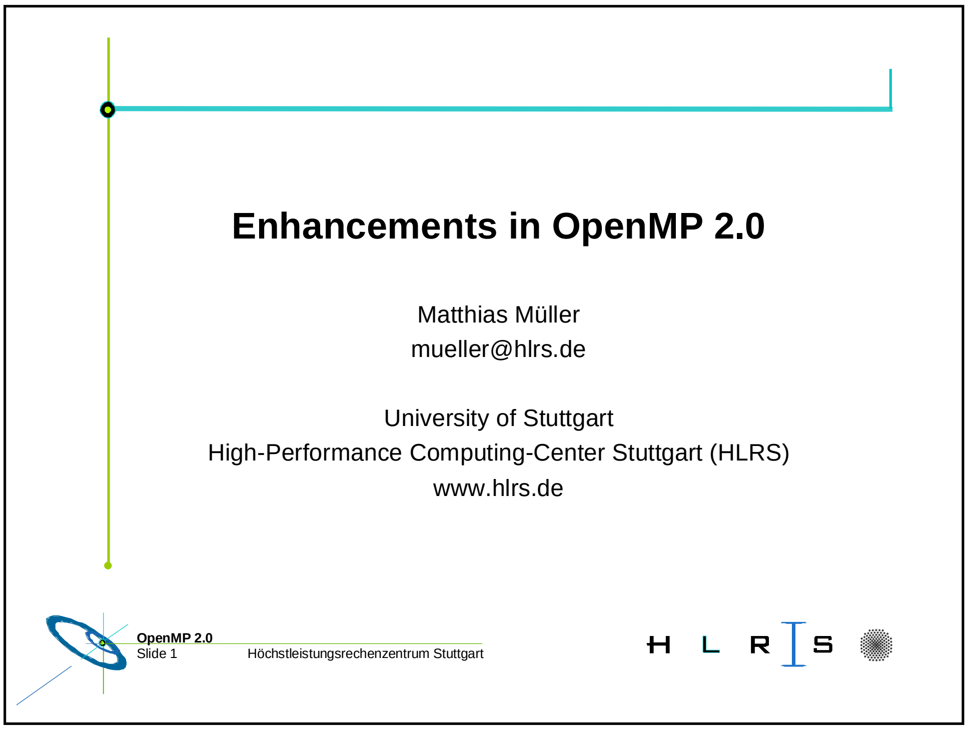
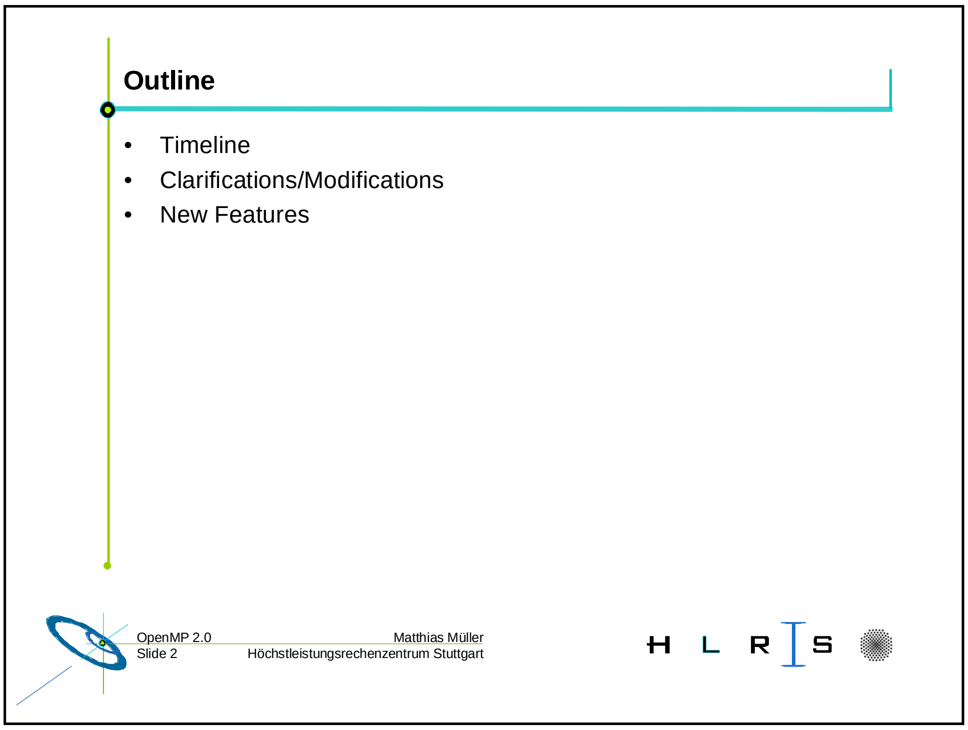


Enhancements in OpenMP 2.0

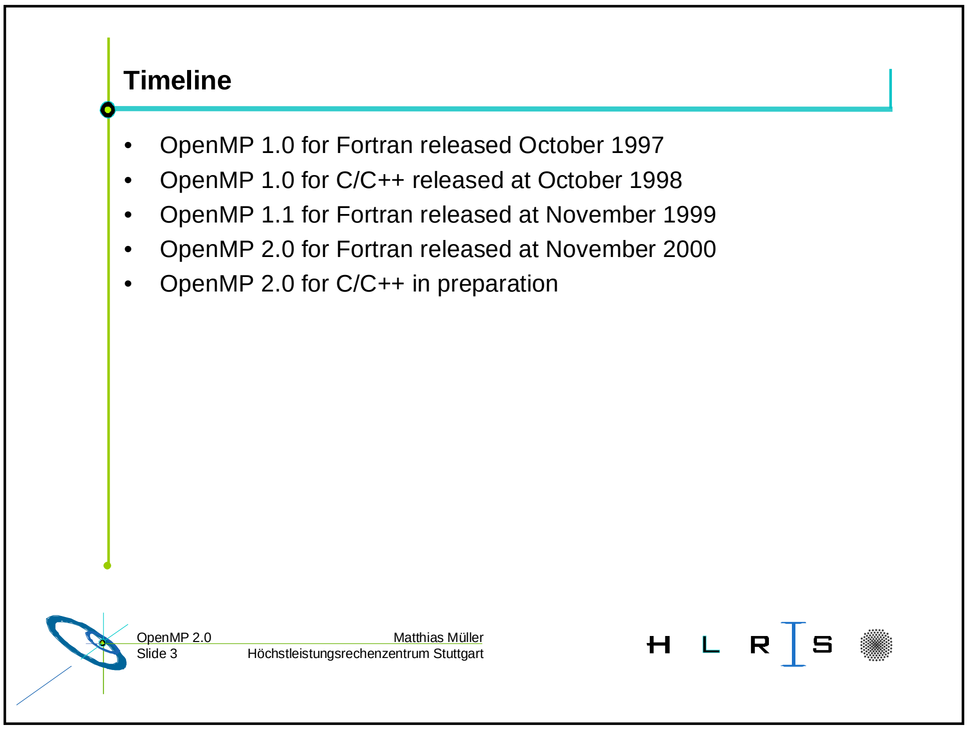
Matthias Müller
mueller@hlrs.de

University of Stuttgart
High-Performance Computing-Center Stuttgart (HLRS)
www.hlrs.de



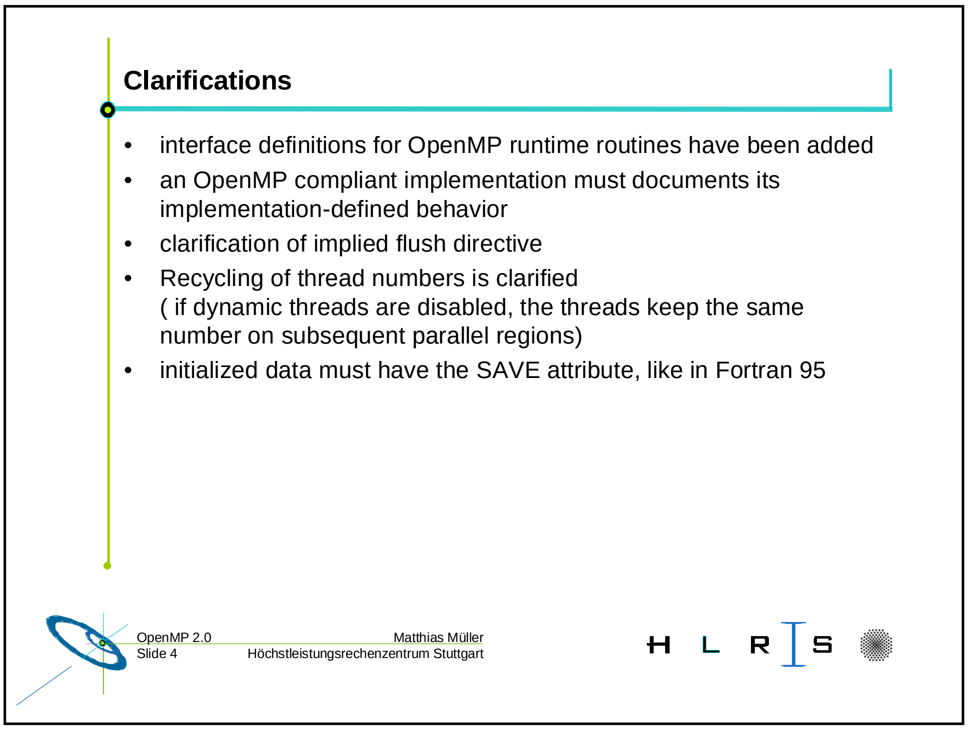
Outline

- Timeline
- Clarifications/Modifications
- New Features



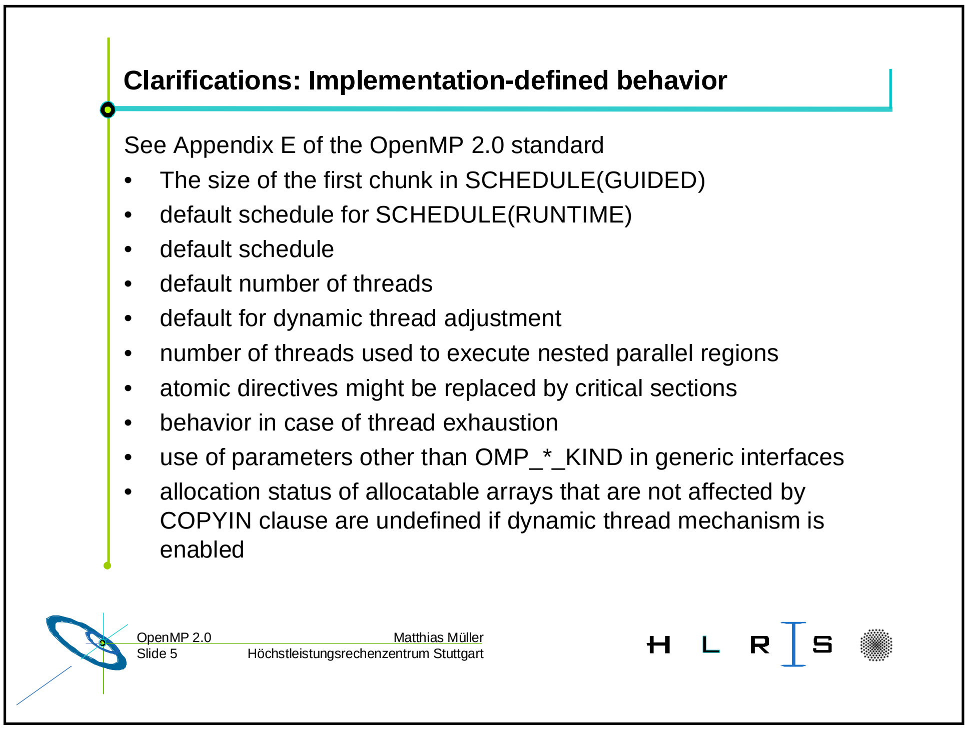
Timeline

- OpenMP 1.0 for Fortran released October 1997
- OpenMP 1.0 for C/C++ released at October 1998
- OpenMP 1.1 for Fortran released at November 1999
- OpenMP 2.0 for Fortran released at November 2000
- OpenMP 2.0 for C/C++ in preparation



Clarifications

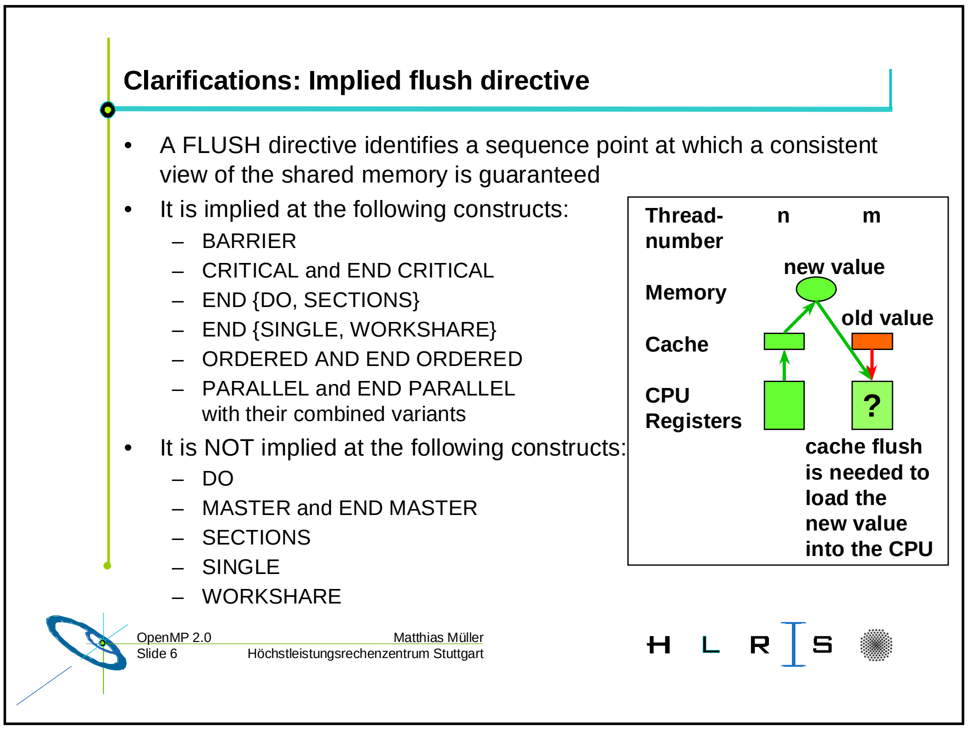
- interface definitions for OpenMP runtime routines have been added
- an OpenMP compliant implementation must documents its implementation-defined behavior
- clarification of implied flush directive
- Recycling of thread numbers is clarified
  (if dynamic threads are disabled, the threads keep the same number on subsequent parallel regions)
- initialized data must have the SAVE attribute, like in Fortran 95



Clarifications: Implementation-defined behavior

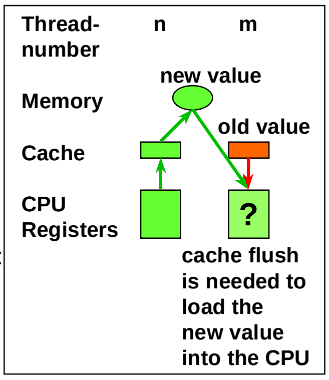
See Appendix E of the OpenMP 2.0 standard

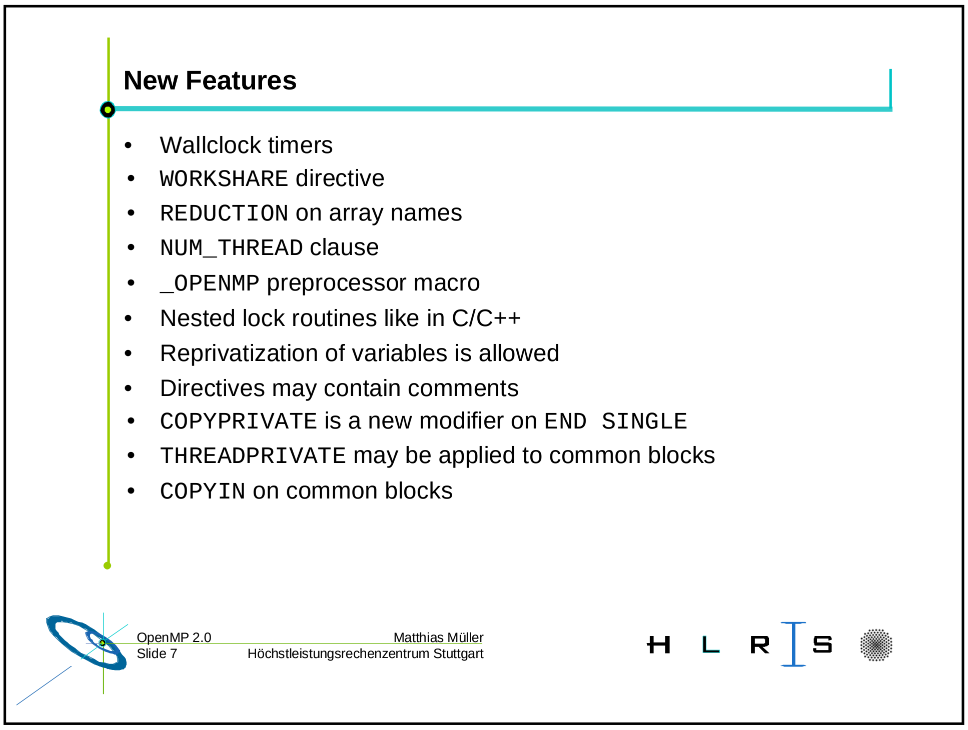
- The size of the first chunk in SCHEDULE(GUIDED)
- default schedule for SCHEDULE(RUNTIME)
- default schedule
- default number of threads
- default for dynamic thread adjustment
- number of threads used to execute nested parallel regions
- atomic directives might be replaced by critical sections
- behavior in case of thread exhaustion
- use of parameters other than OMP_*_KIND in generic interfaces
- allocation status of allocatable arrays that are not affected by COPYIN clause are undefined if dynamic thread mechanism is enabled



Clarifications: Implied flush directive

- A FLUSH directive identifies a sequence point at which a consistent view of the shared memory is guaranteed
- It is implied at the following constructs:
  - BARRIER
  - CRITICAL and END CRITICAL
  - END {DO, SECTIONS}
  - END {SINGLE, WORKSHARE}
  - ORDERED AND END ORDERED
  - PARALLEL and END PARALLEL with their combined variants
- It is NOT implied at the following constructs:
  - DO
  - MASTER and END MASTER
  - SECTIONS
  - SINGLE
  - WORKSHARE

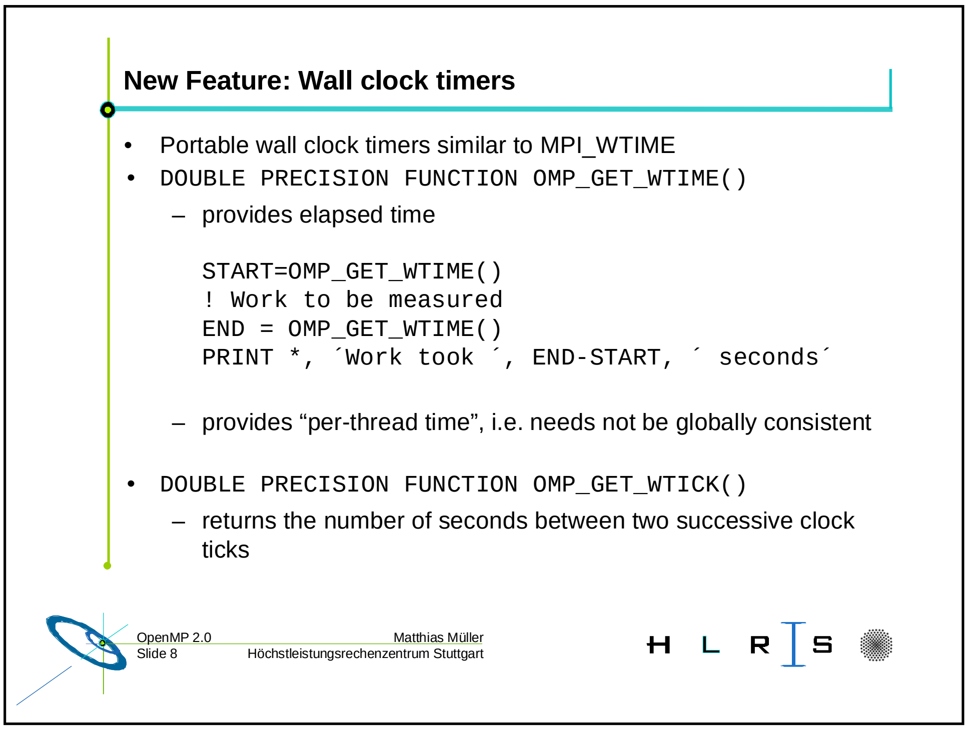




New Features

- Wallclock timers
- `WORKSHARE` directive
- `REDUCTION` on array names
- `NUM_THREAD` clause
- `_OPENMP` preprocessor macro
- Nested lock routines like in C/C++
- Reprivatization of variables is allowed
- Directives may contain comments
- `COPYPRIVATE` is a new modifier on `END SINGLE`
- `THREADPRIVATE` may be applied to common blocks
- `COPYIN` on common blocks

OpenMP 2.0
Slide 7
Matthias Müller
Höchstleistungsrechenzentrum Stuttgart



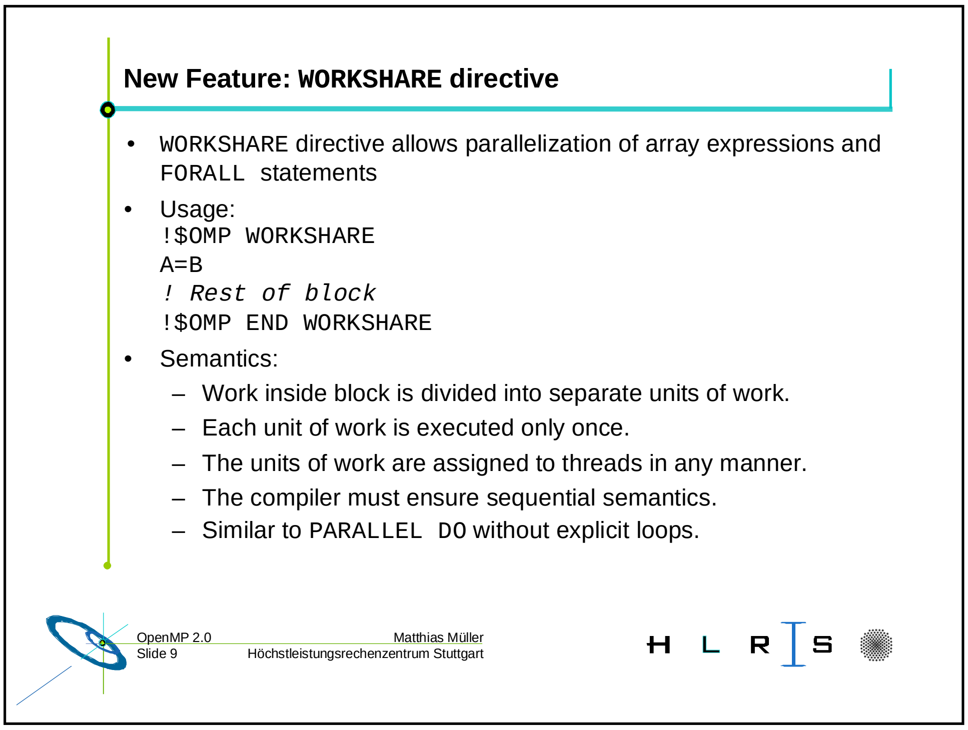
New Feature: Wall clock timers

- Portable wall clock timers similar to MPI_WTIME
- `DOUBLE PRECISION FUNCTION OMP_GET_WTIME()`
  - provides elapsed time

```
START=OMP_GET_WTIME()
! Work to be measured
END = OMP_GET_WTIME()
PRINT *, ´Work took ´, END-START, ´ seconds´
```

  - provides “per-thread time”, i.e. needs not be globally consistent
- `DOUBLE PRECISION FUNCTION OMP_GET_WTICK()`
  - returns the number of seconds between two successive clock ticks

OpenMP 2.0
Slide 8
Matthias Müller
Höchstleistungsrechenzentrum Stuttgart



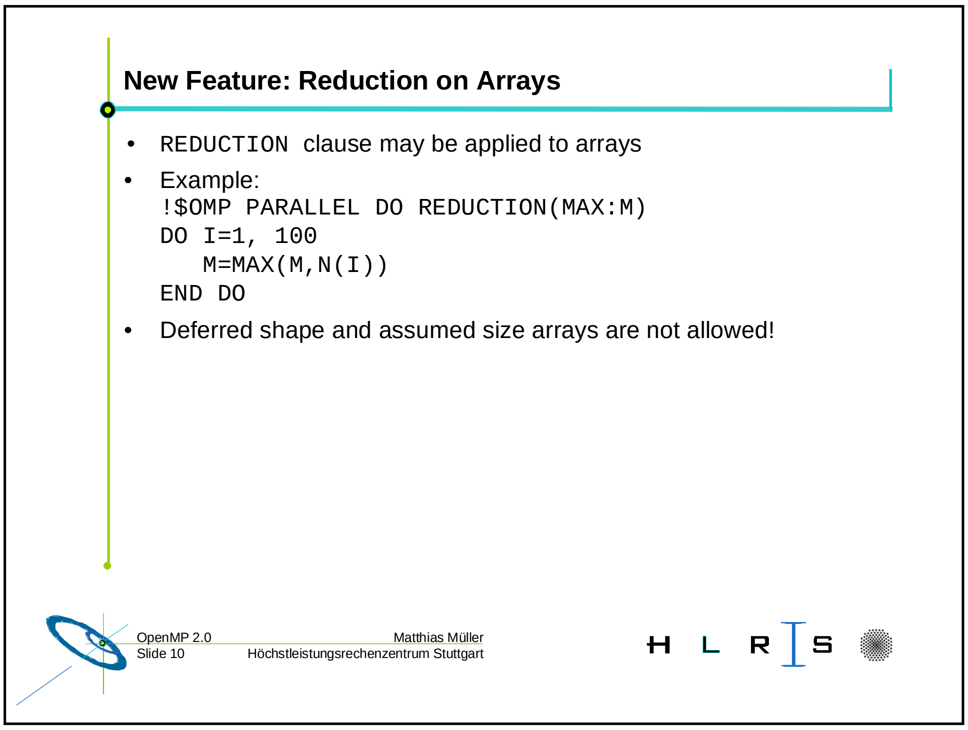
New Feature: WORKSHARE directive

- `WORKSHARE` directive allows parallelization of array expressions and `FORALL` statements
- Usage:

```
!$OMP WORKSHARE
A=B
! Rest of block
!$OMP END WORKSHARE
```

- Semantics:
  – Work inside block is divided into separate units of work.
  – Each unit of work is executed only once.
  – The units of work are assigned to threads in any manner.
  – The compiler must ensure sequential semantics.
  – Similar to `PARALLEL DO` without explicit loops.

OpenMP 2.0 Slide 9 Matthias Müller Höchstleistungsrechenzentrum Stuttgart



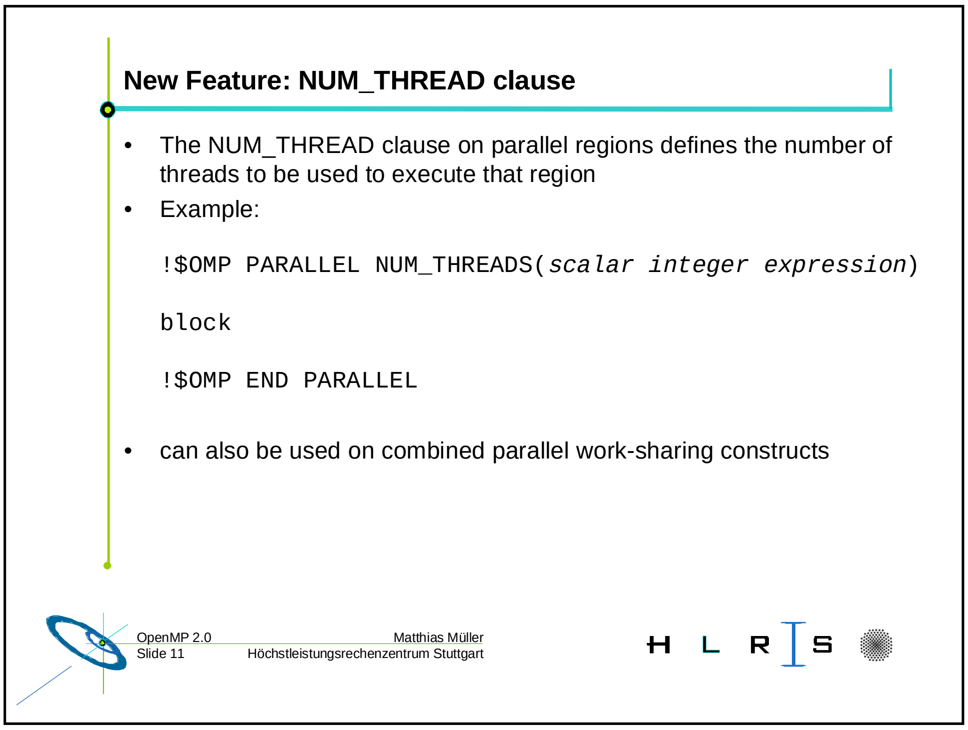
New Feature: Reduction on Arrays

- `REDUCTION` clause may be applied to arrays
- Example:

```
!$OMP PARALLEL DO REDUCTION(MAX:M)
DO I=1, 100
   M=MAX(M,N(I))
END DO
```

- Deferred shape and assumed size arrays are not allowed!

OpenMP 2.0 Slide 10 Matthias Müller Höchstleistungsrechenzentrum Stuttgart



New Feature: NUM_THREAD clause

- The NUM_THREAD clause on parallel regions defines the number of threads to be used to execute that region
- Example:

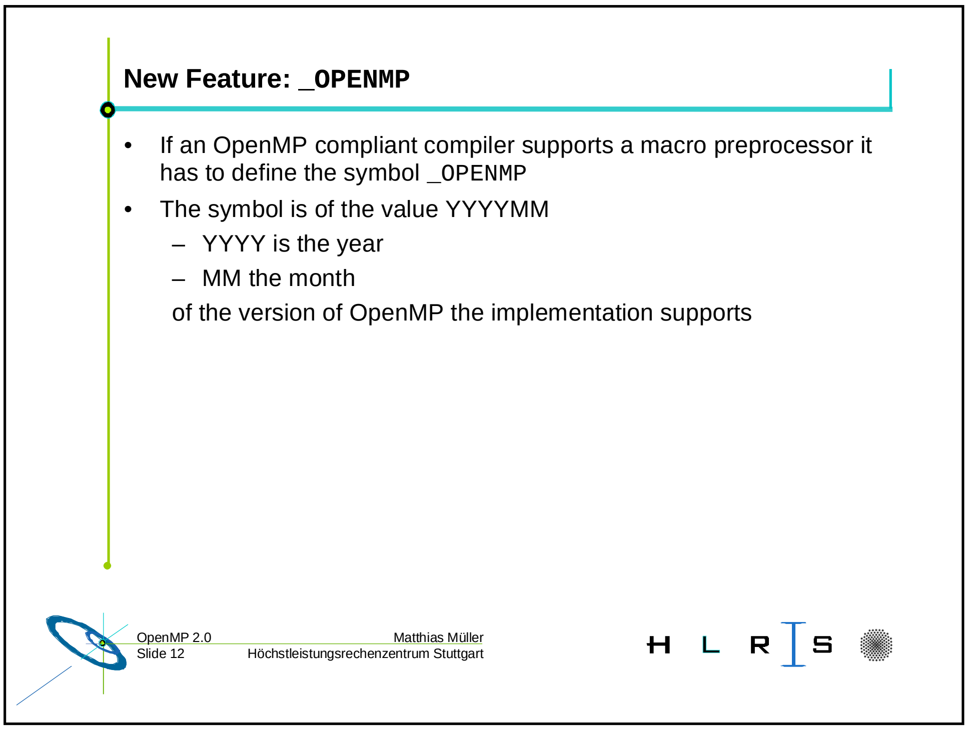
```
!$OMP PARALLEL NUM_THREADS(scalar integer expression)

block

!$OMP END PARALLEL
```

- can also be used on combined parallel work-sharing constructs

OpenMP 2.0 Matthias Müller
Slide 11 Höchstleistungsrechenzentrum Stuttgart

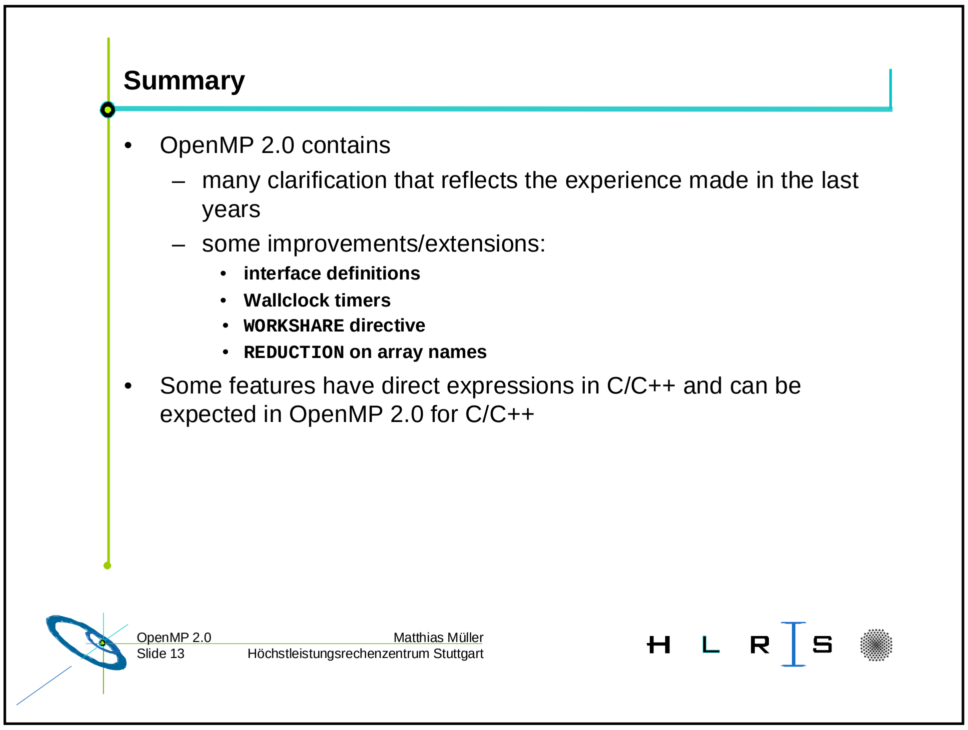


New Feature: _OPENMP

- If an OpenMP compliant compiler supports a macro preprocessor it has to define the symbol `_OPENMP`
- The symbol is of the value YYYYMM
  - YYYY is the year
  - MM the month

  of the version of OpenMP the implementation supports

OpenMP 2.0 Matthias Müller
Slide 12 Höchstleistungsrechenzentrum Stuttgart



Summary

- OpenMP 2.0 contains
  - many clarification that reflects the experience made in the last years
  - some improvements/extensions:
    - **interface definitions**
    - **Wallclock timers**
    - **`WORKSHARE` directive**
    - **`REDUCTION` on array names**
- Some features have direct expressions in C/C++ and can be expected in OpenMP 2.0 for C/C++

OpenMP 2.0
Slide 13

Matthias Müller
Höchstleistungsrechenzentrum Stuttgart

H L R I S