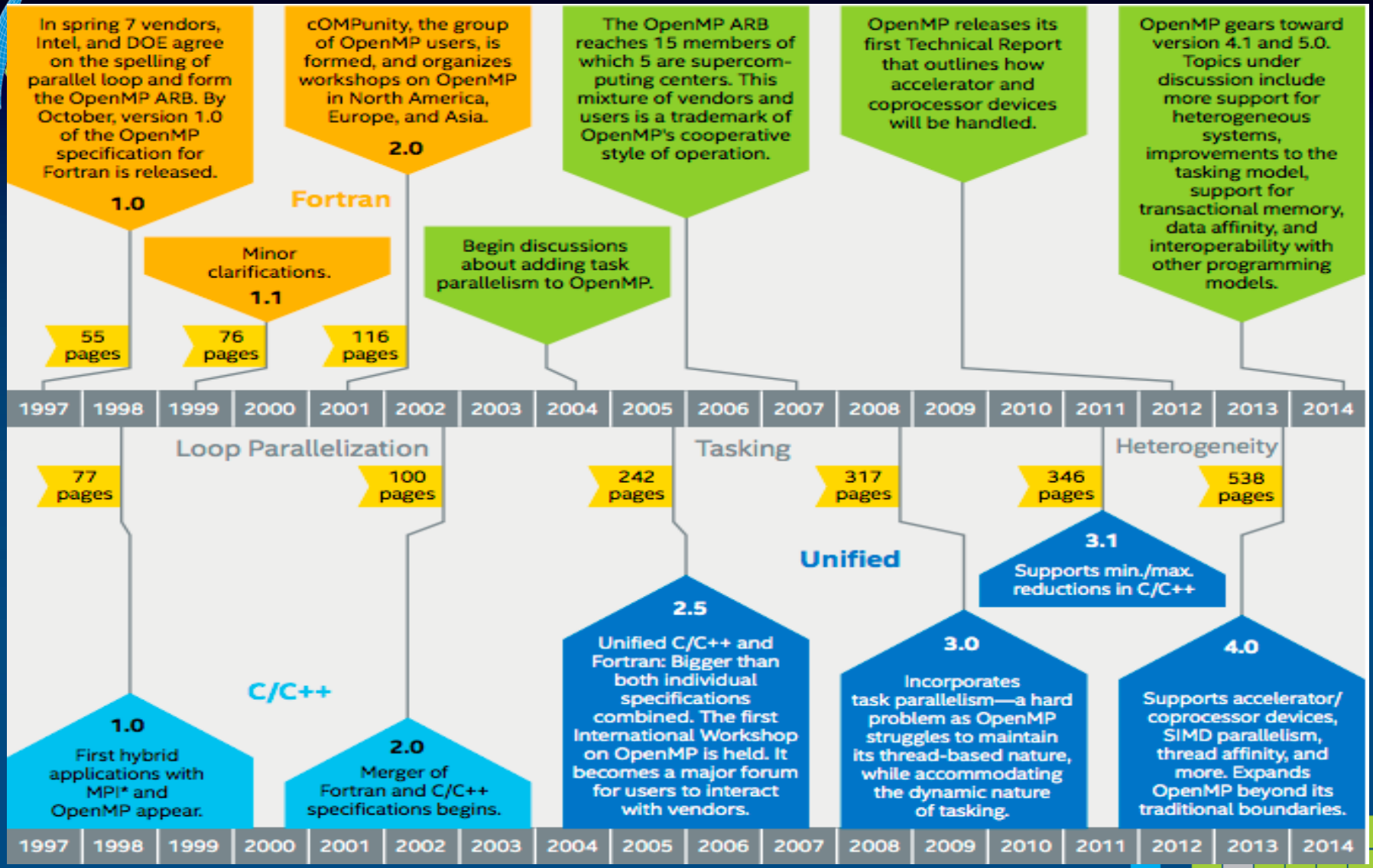
Introduction to OpenMP

ICHEC
Irish Centre for High-End Computing

ICHEC
Irish Centre for High-End Computing

What is OpenMP?

- Open specifications for Multi Processing.
- API to explicitly specify multi-threaded, shared-memory parallelism.
- Developed by a consortium of computer vendors for portable and scalable parallel programming, multi-platform.
- Flexible and easy to implement.
- Three primary API components:
  - Compiler directives
  - Runtime library routines
  - Environment variables
- Designed for C, C++ and Fortran.



- 4.5 on Nov 2015, 5.0 on Nov 2018
- https://software.intel.com/sites/default/files/managed/6a/78/parallel_mag_issue18.pdf

ICHEC
Irish Centre for High-End Computing

Basic OpenMP Terminology

- **thread**: a lightweight process - an instance of the program and data.
- **thread team**: a set of threads co-operating on a task; master+workers
- **master thread**: the coordinating thread.

- **directive/pragma**: pre-processed OpenMP code.

- **construct**: an OpenMP executable directive.
- **region:** dynamic or runtime extent of a construct or of an OpenMP library routine.
- **clause**: controls the operation of the directive or data.

ICHEC
Irish Centre for High-End Computing

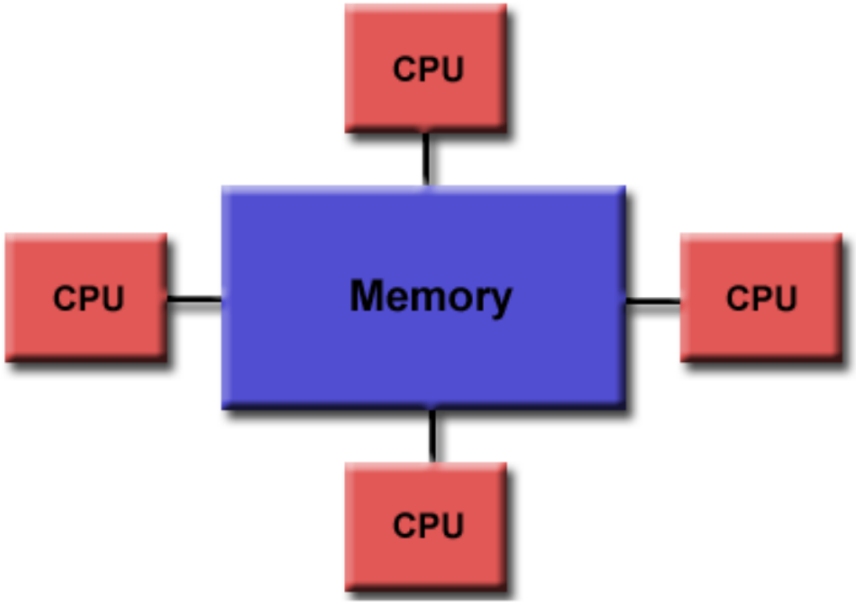
Thread Safety

- **Thread safe**: program is executed as intended when multiple threads are involved.
- **Thread unsafe**: data should not be accessed simultaneously by multiple threads.
- **Race condition**: execution of the program depends on the order in which threads complete their tasks.
- **Deadlock**: two (or more) threads mutually require each other to complete a task to proceed.

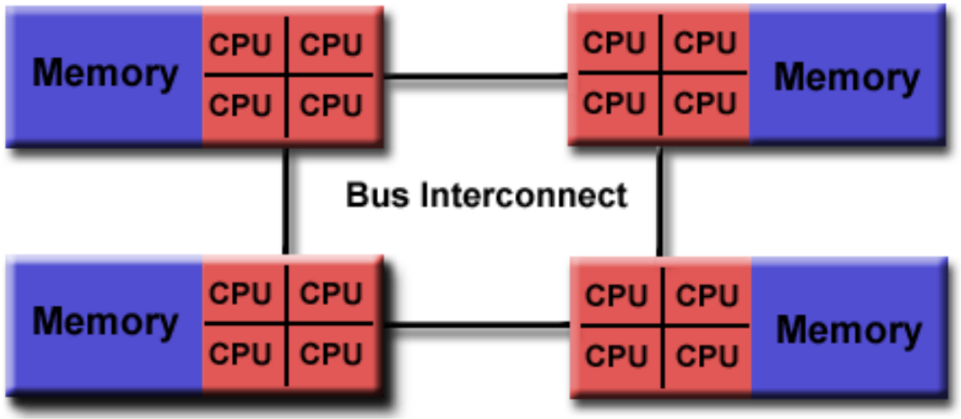
ICHEC
Irish Centre for High-End Computing

OpenMP Programming Model

- Multi-processor/core systems.
- Shared-memory UMA or NUMA systems.



Uniform Memory Access



Non-Uniform Memory Access

ICHEC
Irish Centre for High-End Computing

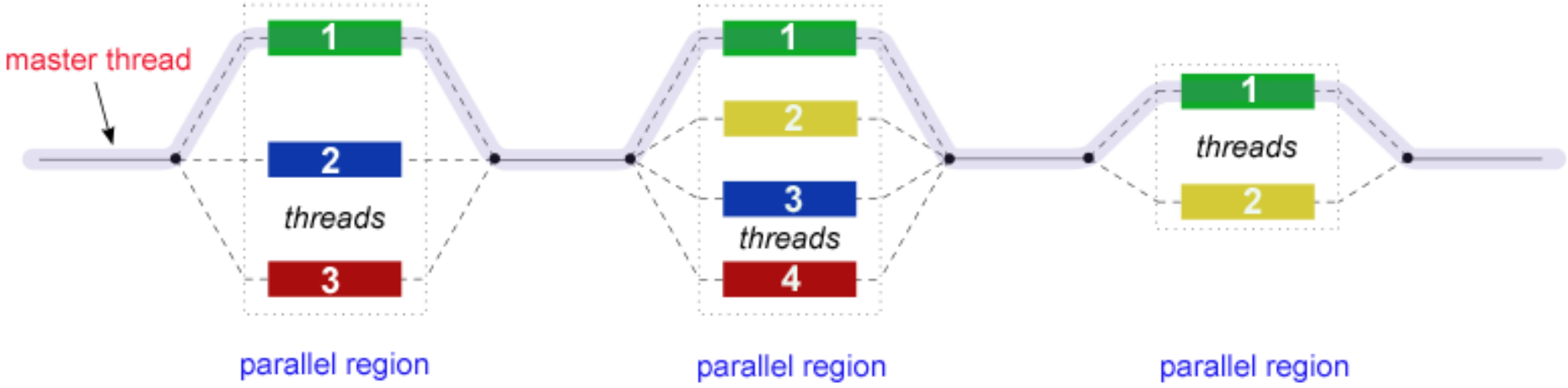
OpenMP Execution Model … (1)

- Thread Based Parallelism
  - Execute a block of code with multiple threads.
  - Rule of thumb: One thread per core.
- Compiler Directive Based
  - User inserts directives telling compiler how to execute statements.
- Explicit Parallelism
  - OpenMP directives, library functions and environment variables explicitly specify what and how to parallelise a block of code.

ICHEC
Irish Centre for High-End Computing

OpenMP Execution Model … (2)

- Fork-Join Model

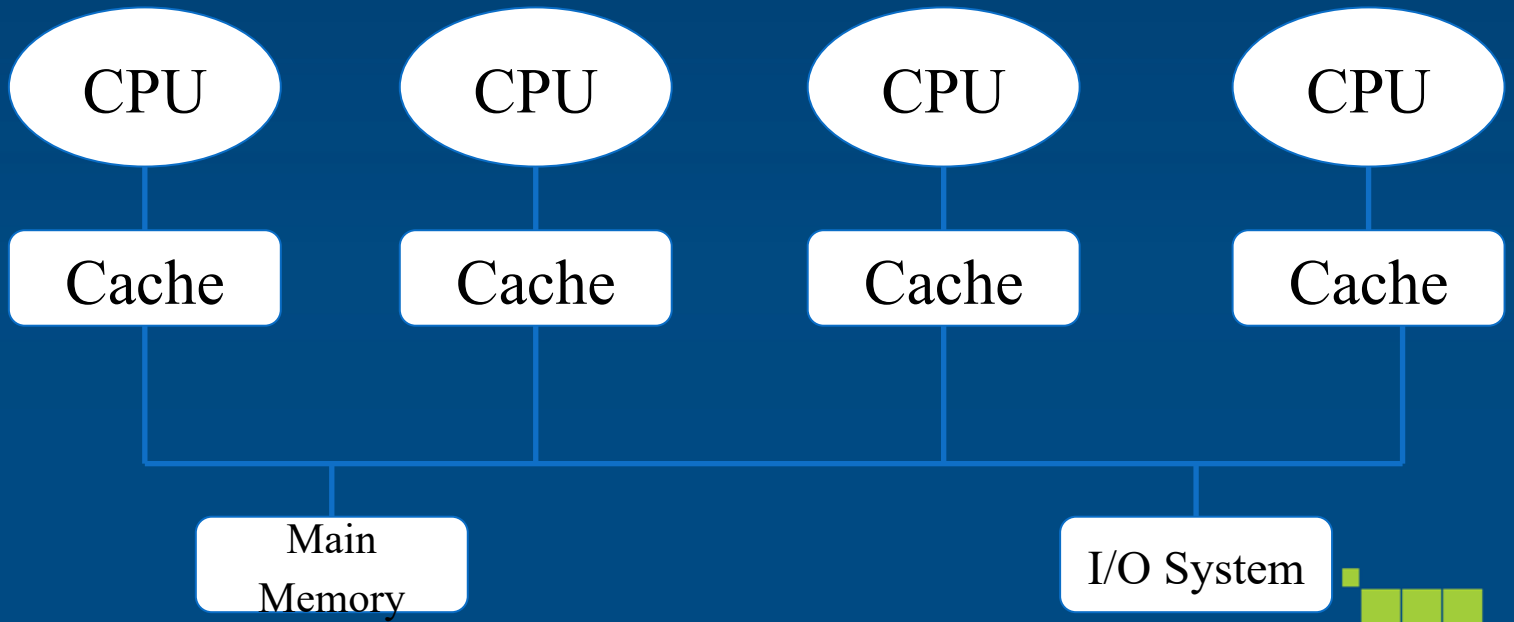


- Nested parallelism
  - Placement of parallel regions inside parallel regions
- Dynamic threads
  - Runtime environment can dynamically alter number of threads used to execute a parallel region.

ICHEC
Irish Centre for High-End Computing

OpenMP Memory Model … (1)

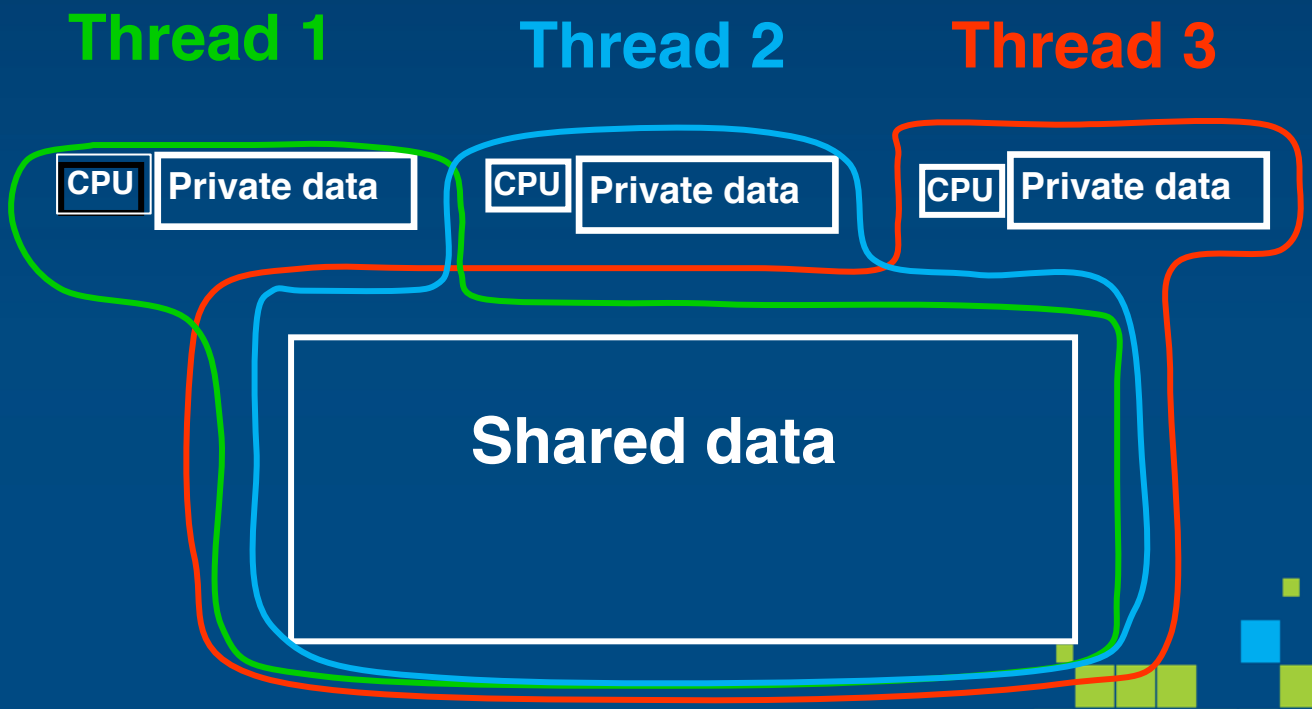
- All threads have access to the shared memory.
- Threads communicate by sharing variables
- If multiple threads write the same memory unit then a data race occurs. Use Synchronization to protect data conflicts.
- The OpenMP flush provides a consistent view of memory



ICHEC
Irish Centre for High-End Computing

OpenMP Memory Model … (2)

- Data can be shared or private.
- Shared data is accessible by all threads.
- Private data can only be accessed by the thread that owns it.



ICHEC
Irish Centre for High-End Computing

OpenMP Components

- Compiler Directives
  - Specify a parallel region
  - Divide work among threads
  - Synchronize threads
- Runtime Library Functions
  - Set and query thread-related information
    - number of threads, identifier, team size, …
  - Query if in parallel region
  - Implement and coordinate locks
  - Query timing-related information
    - wall-clock time and resolution
- Environment Variables
  - Specify number of threads
  - Divide work among threads
  - Bind threads to processors

ICHEC
Irish Centre for High-End Computing

Compiling OpenMP:

Intel	OpenMP 3.1 fully supported in version 12.0, 13.0, 14.0 compilers OpenMP 4.0 supported in version 15.0 and 16.0 compilers OpenMP 4.5 supported in version 17.0, 18.0 and 19.0 compilers OpenMP 5.0 C/C++/Fortran supported in 19.1 compilers. *Compile with -openmp or -qopenmp to enable OpenMP.* *icc (C) icpc (C++) ifort (Fortran)*
GNU	OpenMP 2.5 is fully supported from GCC 4.2.0 OpenMP 3.0 is fully supported from GCC 4.4.0 OpenMP 3.1 is fully supported from GCC 4.7.0 OpenMP 4.0 is supported in GCC 4.9.0 (only C and C++) OpenMP 4.0 is fully supported from GCC 4.9.1 OpenMP 4.5 is fully supported from GCC 6.1 (only C and C++) OpenMP 5.0 is partially supported from GCC 9.1 (only C and C++) Compile with -fopenmp to enable OpenMP gcc (C) g++ (C++) g77/gfortran (Fortran)

https://www.openmp.org/resources/openmp-compilers-tools/

ICHEC
Irish Centre for High-End Computing

Approaches to Parallelism

- Parallel-regions Parallelism:
  - any sections of codes can be parallelized
  - using the thread identifier to distribute the work
  - coarse-grained parallelism
- Loop-level Parallelism:
  - individual loops parallelized
  - each thread assigned a unique range of the loop index
  - fine-grained parallelism
- Task-level Parallelism: Supported after v3.0