OpenMP

From Wikipedia, the free encyclopedia

[Jump to navigation](https://en.wikipedia.org/wiki/OpenMP#mw-head)[Jump to search](https://en.wikipedia.org/wiki/OpenMP#searchInput)

*Not to be confused with*[*Open MPI*](https://en.wikipedia.org/wiki/Open_MPI)*.*

|  |  |
| --- | --- |
| **OpenMP** | |
| [OpenMP logo](https://en.wikipedia.org/wiki/File:OpenMP_logo.png) | |
| [**Original author(s)**](https://en.wikipedia.org/wiki/Software_developer) | OpenMP Architecture Review Board[[1]](https://en.wikipedia.org/wiki/OpenMP#cite_note-Board-1) |
| [**Developer(s)**](https://en.wikipedia.org/wiki/Software_developer) | OpenMP Architecture Review Board[[1]](https://en.wikipedia.org/wiki/OpenMP#cite_note-Board-1) |
| [**Stable release**](https://en.wikipedia.org/wiki/Software_release_life_cycle) | 5.0 / November 8, 2018; 21 months ago |
| [**Operating system**](https://en.wikipedia.org/wiki/Operating_system) | [Cross-platform](https://en.wikipedia.org/wiki/Cross-platform) |
| [**Platform**](https://en.wikipedia.org/wiki/Computing_platform) | Cross-platform |
| [**Type**](https://en.wikipedia.org/wiki/Software_categories#Categorization_approaches) | Extension to [C](https://en.wikipedia.org/wiki/C_(programming_language)), [C++](https://en.wikipedia.org/wiki/C%2B%2B), and [Fortran](https://en.wikipedia.org/wiki/Fortran); [API](https://en.wikipedia.org/wiki/Application_programming_interface) |
| [**License**](https://en.wikipedia.org/wiki/Software_license) | Various[[2]](https://en.wikipedia.org/wiki/OpenMP#cite_note-openmp.org-2) |
| **Website** | [openmp.org](http://openmp.org/) |

The [application programming interface](https://en.wikipedia.org/wiki/Application_programming_interface) (API) **OpenMP** (**Open Multi-Processing**) supports multi-platform [shared-memory](https://en.wikipedia.org/wiki/Shared_memory_architecture) [multiprocessing](https://en.wikipedia.org/wiki/Multiprocessing) programming in [C](https://en.wikipedia.org/wiki/C_(programming_language)), [C++](https://en.wikipedia.org/wiki/C%2B%2B), and [Fortran](https://en.wikipedia.org/wiki/Fortran),[[3]](https://en.wikipedia.org/wiki/OpenMP#cite_note-OSConcepts-3) on many platforms, [instruction-set architectures](https://en.wikipedia.org/wiki/Instruction_set_architecture) and [operating systems](https://en.wikipedia.org/wiki/Operating_system), including [Solaris](https://en.wikipedia.org/wiki/Solaris_(operating_system)), [AIX](https://en.wikipedia.org/wiki/IBM_AIX), [HP-UX](https://en.wikipedia.org/wiki/HP-UX), [Linux](https://en.wikipedia.org/wiki/Linux), [macOS](https://en.wikipedia.org/wiki/MacOS), and [Windows](https://en.wikipedia.org/wiki/Microsoft_Windows). It consists of a set of [compiler directives](https://en.wikipedia.org/wiki/Compiler_directive), [library routines](https://en.wikipedia.org/wiki/Library_(computing)), and [environment variables](https://en.wikipedia.org/wiki/Environment_variable) that influence run-time behavior.[[2]](https://en.wikipedia.org/wiki/OpenMP#cite_note-openmp.org-2)[[4]](https://en.wikipedia.org/wiki/OpenMP#cite_note-4)[[5]](https://en.wikipedia.org/wiki/OpenMP#cite_note-5)

OpenMP is managed by the [nonprofit](https://en.wikipedia.org/wiki/Nonprofit_organization) technology [consortium](https://en.wikipedia.org/wiki/Consortium) *OpenMP Architecture Review Board* (or *OpenMP ARB*), jointly defined by a group of major computer hardware and software vendors, including [Arm](https://en.wikipedia.org/wiki/Arm_Holdings), [AMD](https://en.wikipedia.org/wiki/AMD), [IBM](https://en.wikipedia.org/wiki/IBM), [Intel](https://en.wikipedia.org/wiki/Intel), [Cray](https://en.wikipedia.org/wiki/Cray), [HP](https://en.wikipedia.org/wiki/Hewlett-Packard), [Fujitsu](https://en.wikipedia.org/wiki/Fujitsu), [Nvidia](https://en.wikipedia.org/wiki/Nvidia), [NEC](https://en.wikipedia.org/wiki/NEC), [Red Hat](https://en.wikipedia.org/wiki/Red_Hat), [Texas Instruments](https://en.wikipedia.org/wiki/Texas_Instruments), [Oracle Corporation](https://en.wikipedia.org/wiki/Oracle_Corporation), and more.[[1]](https://en.wikipedia.org/wiki/OpenMP#cite_note-Board-1)

OpenMP uses a [portable](https://en.wikipedia.org/wiki/Software_portability), scalable model that gives [programmers](https://en.wikipedia.org/wiki/Programmer) a simple and flexible interface for developing parallel applications for platforms ranging from the standard [desktop computer](https://en.wikipedia.org/wiki/Desktop_computer) to the [supercomputer](https://en.wikipedia.org/wiki/Supercomputer).

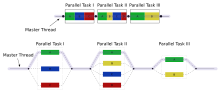
An application built with the hybrid model of [parallel programming](https://en.wikipedia.org/wiki/Parallel_programming) can run on a [computer cluster](https://en.wikipedia.org/wiki/Computer_cluster) using both OpenMP and [Message Passing Interface](https://en.wikipedia.org/wiki/Message_Passing_Interface) (MPI), such that OpenMP is used for parallelism *within* a (multi-core) node while MPI is used for parallelism *between* nodes. There have also been efforts to run OpenMP on [software distributed shared memory](https://en.wikipedia.org/wiki/Distributed_shared_memory) systems,[[6]](https://en.wikipedia.org/wiki/OpenMP#cite_note-6) to translate OpenMP into MPI[[7]](https://en.wikipedia.org/wiki/OpenMP#cite_note-7)[[8]](https://en.wikipedia.org/wiki/OpenMP#cite_note-8) and to extend OpenMP for non-shared memory systems.[[9]](https://en.wikipedia.org/wiki/OpenMP#cite_note-9)



**Contents**

* [1Design](https://en.wikipedia.org/wiki/OpenMP#Design)
* [2History](https://en.wikipedia.org/wiki/OpenMP#History)
* [3Core elements](https://en.wikipedia.org/wiki/OpenMP#Core_elements)
  + [3.1Thread creation](https://en.wikipedia.org/wiki/OpenMP#Thread_creation)
  + [3.2Work-sharing constructs](https://en.wikipedia.org/wiki/OpenMP#Work-sharing_constructs)
  + [3.3Variant directives](https://en.wikipedia.org/wiki/OpenMP#Variant_directives)
  + [3.4Clauses](https://en.wikipedia.org/wiki/OpenMP#Clauses)
  + [3.5User-level runtime routines](https://en.wikipedia.org/wiki/OpenMP#User-level_runtime_routines)
  + [3.6Environment variables](https://en.wikipedia.org/wiki/OpenMP#Environment_variables)
* [4Implementations](https://en.wikipedia.org/wiki/OpenMP#Implementations)
* [5Pros and cons](https://en.wikipedia.org/wiki/OpenMP#Pros_and_cons)
* [6Performance expectations](https://en.wikipedia.org/wiki/OpenMP#Performance_expectations)
* [7Thread affinity](https://en.wikipedia.org/wiki/OpenMP#Thread_affinity)
* [8Benchmarks](https://en.wikipedia.org/wiki/OpenMP#Benchmarks)
* [9See also](https://en.wikipedia.org/wiki/OpenMP#See_also)
* [10References](https://en.wikipedia.org/wiki/OpenMP#References)
* [11Further reading](https://en.wikipedia.org/wiki/OpenMP#Further_reading)
* [12External links](https://en.wikipedia.org/wiki/OpenMP#External_links)

Design[[edit](https://en.wikipedia.org/w/index.php?title=OpenMP&action=edit&section=1)]

[](https://en.wikipedia.org/wiki/File:Fork_join.svg)

An illustration of [multithreading](https://en.wikipedia.org/wiki/Thread_(computer_science)) where the primary thread forks off a number of threads which execute blocks of code in parallel.

*See also:*[*Fork–join model*](https://en.wikipedia.org/wiki/Fork%E2%80%93join_model)

OpenMP is an implementation of [multithreading](https://en.wikipedia.org/wiki/Thread_(computer_science)), a method of parallelizing whereby a *primary* thread (a series of instructions executed consecutively) [*forks*](https://en.wikipedia.org/wiki/Fork_(system_call)) a specified number of *sub*-threads and the system divides a task among them. The threads then run [concurrently](https://en.wikipedia.org/wiki/Concurrent_computing), with the [runtime environment](https://en.wikipedia.org/wiki/Runtime_environment) allocating threads to different processors.

The section of code that is meant to run in parallel is marked accordingly, with a compiler directive that will cause the threads to form before the section is executed.[[3]](https://en.wikipedia.org/wiki/OpenMP#cite_note-OSConcepts-3) Each thread has an *id* attached to it which can be obtained using a [function](https://en.wikipedia.org/wiki/Function_(computer_science)) (called omp\_get\_thread\_num()). The thread id is an integer, and the primary thread has an id of *0*. After the execution of the parallelized code, the threads *join* back into the primary thread, which continues onward to the end of the program.

By default, each thread executes the parallelized section of code independently. *Work-sharing constructs* can be used to divide a task among the threads so that each thread executes its allocated part of the code. Both [task parallelism](https://en.wikipedia.org/wiki/Task_parallelism) and [data parallelism](https://en.wikipedia.org/wiki/Data_parallelism) can be achieved using OpenMP in this way.

The runtime environment allocates threads to processors depending on usage, machine load and other factors. The runtime environment can assign the number of threads based on [environment variables](https://en.wikipedia.org/wiki/Environment_variable), or the code can do so using functions. The OpenMP functions are included in a [header file](https://en.wikipedia.org/wiki/Header_file) labelled omp.h in [C](https://en.wikipedia.org/wiki/C_(programming_language))/[C++](https://en.wikipedia.org/wiki/C%2B%2B).

History[[edit](https://en.wikipedia.org/w/index.php?title=OpenMP&action=edit&section=2)]

The OpenMP Architecture Review Board (ARB) published its first API specifications, OpenMP for Fortran 1.0, in October 1997. In October the following year they released the C/C++ standard. 2000 saw version 2.0 of the Fortran specifications with version 2.0 of the C/C++ specifications being released in 2002. Version 2.5 is a combined C/C++/Fortran specification that was released in 2005.

Up to version 2.0, OpenMP primarily specified ways to parallelize highly regular loops, as they occur in matrix-oriented [numerical programming](https://en.wikipedia.org/wiki/Numerical_programming), where the number of iterations of the loop is known at entry time. This was recognized as a limitation, and various task parallel extensions were added to implementations. In 2005, an effort to standardize task parallelism was formed, which published a proposal in 2007, taking inspiration from task parallelism features in [Cilk](https://en.wikipedia.org/wiki/Cilk), [X10](https://en.wikipedia.org/wiki/X10_(programming_language)) and [Chapel](https://en.wikipedia.org/wiki/Chapel_(programming_language)).[[10]](https://en.wikipedia.org/wiki/OpenMP#cite_note-10)

Version 3.0 was released in May 2008. Included in the new features in 3.0 is the concept of *tasks* and the *task* construct,[[11]](https://en.wikipedia.org/wiki/OpenMP#cite_note-11) significantly broadening the scope of OpenMP beyond the parallel loop constructs that made up most of OpenMP 2.0.[[12]](https://en.wikipedia.org/wiki/OpenMP#cite_note-12)

Version 4.0 of the specification was released in July 2013.[[13]](https://en.wikipedia.org/wiki/OpenMP#cite_note-13) It adds or improves the following features: support for [accelerators](https://en.wikipedia.org/wiki/Hardware_acceleration); [atomics](https://en.wikipedia.org/wiki/Linearizability); error handling; [thread affinity](https://en.wikipedia.org/wiki/Processor_affinity); tasking extensions; user defined [reduction](https://en.wikipedia.org/wiki/Fold_(higher-order_function)); [SIMD](https://en.wikipedia.org/wiki/SIMD) support; [Fortran 2003](https://en.wikipedia.org/wiki/Fortran_2003) support.[[14]](https://en.wikipedia.org/wiki/OpenMP#cite_note-14)[[*full citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citing_sources#What_information_to_include)]

The current version is 5.0, released in November 2018.

Note that not all compilers (and OSes) support the full set of features for the latest version/s.

Core elements[[edit](https://en.wikipedia.org/w/index.php?title=OpenMP&action=edit&section=3)]

[](https://en.wikipedia.org/wiki/File:OpenMP_language_extensions.svg)

Chart of OpenMP constructs

The core elements of OpenMP are the constructs for thread creation, workload distribution (work sharing), data-environment management, thread synchronization, user-level runtime routines and environment variables.

In C/C++, OpenMP uses [#pragmas](https://en.wikipedia.org/wiki/C_preprocessor#Compiler-specific_preprocessor_features). The OpenMP specific pragmas are listed below.

**Thread creation**[[edit](https://en.wikipedia.org/w/index.php?title=OpenMP&action=edit&section=4)]

The pragma *omp parallel* is used to fork additional threads to carry out the work enclosed in the construct in parallel. The original thread will be denoted as *master thread* with thread ID 0.

Example (C program): Display "Hello, world." using multiple threads.

#include *<stdio.h>*

#include *<omp.h>*

int main(void)

{

#pragma omp parallel

printf("Hello, world.**\n**");

**return** 0;

}

Use flag -fopenmp to compile using GCC:

$ gcc -fopenmp hello.c -o hello

Output on a computer with two cores, and thus two threads:

Hello, world.

Hello, world.

However, the output may also be garbled because of the [race condition](https://en.wikipedia.org/wiki/Race_condition) caused from the two threads sharing the [standard output](https://en.wikipedia.org/wiki/Standard_output).

Hello, wHello, woorld.

rld.

(Whether printf is thread-safe depends on the implementation. C++ std::cout, on the other hand, is always thread-safe.)

**Work-sharing constructs**[[edit](https://en.wikipedia.org/w/index.php?title=OpenMP&action=edit&section=5)]

Used to specify how to assign independent work to one or all of the threads.

* *omp for* or *omp do*: used to [split up loop iterations](https://en.wikipedia.org/wiki/Map_(parallel_pattern)) among the threads, also called loop constructs.
* *sections*: assigning consecutive but independent code blocks to different threads
* *single*: specifying a code block that is executed by only one thread, a barrier is implied in the end
* *master*: similar to single, but the code block will be executed by the master thread only and no barrier implied in the end.

Example: initialize the value of a large array in parallel, using each thread to do part of the work

int main(int argc, char \*\*argv)

{

int a[100000];

#pragma omp parallel for

**for** (int i = 0; i < 100000; i++) {

a[i] = 2 \* i;

}

**return** 0;

}

This example is [embarrassingly parallel](https://en.wikipedia.org/wiki/Embarrassingly_parallel), and depends only on the value of i. The OpenMP parallel for flag tells the OpenMP system to split this task among its working threads. The threads will each receive a unique and private version of the variable.[[15]](https://en.wikipedia.org/wiki/OpenMP#cite_note-15) For instance, with two worker threads, one thread might be handed a version of i that runs from 0 to 49999 while the second gets a version running from 50000 to 99999.

**Variant directives**[[edit](https://en.wikipedia.org/w/index.php?title=OpenMP&action=edit&section=6)]

Variant directives is one of the major features introduced in OpenMP 5.0 specification to facilitate programmers to improve performance portability. They enable adaptation of OpenMP pragmas and user code at compile time. The specification defines traits to describe active OpenMP constructs, execution devices, and functionality provided by an implementation, context selectors based on the traits and user-defined conditions, and *metadirective* and *declare directive* directives for users to program the same code region with variant directives.

* The *metadirective* is an executable directive that conditionally resolves to another directive at compile time by selecting from multiple directive variants based on traits that define an OpenMP condition or context.
* The *declare variant* directive has similar functionality as *metadirective* but selects a function variant at the call-site based on context or user-defined conditions.

The mechanism provided by the two variant directives for selecting variants is more convenient to use than the C/C++ preprocessing since it directly supports variant selection in OpenMP and allows an OpenMP compiler to analyze and determine the final directive from variants and context.

*// code adaptation using preprocessing directives*

int v1[N], v2[N], v3[N];

#if defined(nvptx)

#pragma omp target teams distribute parallel loop map(to:v1,v2) map(from:v3)

**for** (int i= 0; i< N; i++)

v3[i] = v1[i] \* v2[i];

#else

#pragma omp target parallel loop map(to:v1,v2) map(from:v3)

**for** (int i= 0; i< N; i++)

v3[i] = v1[i] \* v2[i];

#endif

*// code adaptation using metadirective in OpenMP 5.0*

int v1[N], v2[N], v3[N];

#pragma omp target map(to:v1,v2) map(from:v3)

#pragma omp metadirective \

when(device={arch(nvptx)}: target teams distribute parallel loop)\

default(target parallel loop)

**for** (int i= 0; i< N; i++)

v3[i] = v1[i] \* v2[i];

**Clauses**[[edit](https://en.wikipedia.org/w/index.php?title=OpenMP&action=edit&section=7)]

Since OpenMP is a shared memory programming model, most variables in OpenMP code are visible to all threads by default. But sometimes private variables are necessary to avoid [race conditions](https://en.wikipedia.org/wiki/Race_condition) and there is a need to pass values between the sequential part and the parallel region (the code block executed in parallel), so data environment management is introduced as *data sharing attribute clauses* by appending them to the OpenMP directive. The different types of clauses are:

**Data sharing attribute clauses**

* *shared*: the data declared outside a parallel region is shared, which means visible and accessible by all threads simultaneously. By default, all variables in the work sharing region are shared except the loop iteration counter.
* *private*: the data declared within a parallel region is private to each thread, which means each thread will have a local copy and use it as a temporary variable. A private variable is not initialized and the value is not maintained for use outside the parallel region. By default, the loop iteration counters in the OpenMP loop constructs are private.
* *default*: allows the programmer to state that the default data scoping within a parallel region will be either *shared*, or *none* for C/C++, or *shared*, *firstprivate*, *private*, or *none* for Fortran. The *none* option forces the programmer to declare each variable in the parallel region using the data sharing attribute clauses.
* *firstprivate*: like *private* except initialized to original value.
* *lastprivate*: like *private* except original value is updated after construct.
* *reduction*: a safe way of joining work from all threads after construct.

**Synchronization clauses**

* *critical*: the enclosed code block will be executed by only one thread at a time, and not simultaneously executed by multiple threads. It is often used to protect shared data from [race conditions](https://en.wikipedia.org/wiki/Race_condition).
* *atomic*: the memory update (write, or read-modify-write) in the next instruction will be performed atomically. It does not make the entire statement atomic; only the memory update is atomic. A compiler might use special hardware instructions for better performance than when using *critical*.
* *ordered*: the structured block is executed in the order in which iterations would be executed in a sequential loop
* *barrier*: each thread waits until all of the other threads of a team have reached this point. A work-sharing construct has an implicit barrier synchronization at the end.
* *nowait*: specifies that threads completing assigned work can proceed without waiting for all threads in the team to finish. In the absence of this clause, threads encounter a barrier synchronization at the end of the work sharing construct.

**Scheduling clauses**

* *schedule(type, chunk)*: This is useful if the work sharing construct is a do-loop or for-loop. The iteration(s) in the work sharing construct are assigned to threads according to the scheduling method defined by this clause. The three types of scheduling are:

1. *static*: Here, all the threads are allocated iterations before they execute the loop iterations. The iterations are divided among threads equally by default. However, specifying an integer for the parameter *chunk* will allocate chunk number of contiguous iterations to a particular thread.
2. *dynamic*: Here, some of the iterations are allocated to a smaller number of threads. Once a particular thread finishes its allocated iteration, it returns to get another one from the iterations that are left. The parameter *chunk* defines the number of contiguous iterations that are allocated to a thread at a time.
3. *guided*: A large chunk of contiguous iterations are allocated to each thread dynamically (as above). The chunk size decreases exponentially with each successive allocation to a minimum size specified in the parameter *chunk*

**IF control**

* *if*: This will cause the threads to parallelize the task only if a condition is met. Otherwise the code block executes serially.

**Initialization**

* *firstprivate*: the data is private to each thread, but initialized using the value of the variable using the same name from the master thread.
* *lastprivate*: the data is private to each thread. The value of this private data will be copied to a global variable using the same name outside the parallel region if current iteration is the last iteration in the parallelized loop. A variable can be both *firstprivate* and *lastprivate*.
* *threadprivate*: The data is a global data, but it is private in each parallel region during the runtime. The difference between *threadprivate* and *private* is the global scope associated with threadprivate and the preserved value across parallel regions.

**Data copying**

* *copyin*: similar to *firstprivate* for *private* variables, *threadprivate* variables are not initialized, unless using *copyin* to pass the value from the corresponding global variables. No *copyout* is needed because the value of a threadprivate variable is maintained throughout the execution of the whole program.
* *copyprivate*: used with *single* to support the copying of data values from private objects on one thread (the *single* thread) to the corresponding objects on other threads in the team.

**Reduction**

* *reduction(operator | intrinsic : list)*: the variable has a local copy in each thread, but the values of the local copies will be summarized (reduced) into a global shared variable. This is very useful if a particular operation (specified in *operator* for this particular clause) on a variable runs iteratively, so that its value at a particular iteration depends on its value at a prior iteration. The steps that lead up to the operational increment are parallelized, but the threads updates the global variable in a thread safe manner. This would be required in parallelizing [numerical integration](https://en.wikipedia.org/wiki/Numerical_integration) of functions and [differential equations](https://en.wikipedia.org/wiki/Differential_equation), as a common example.

**Others**

* *flush*: The value of this variable is restored from the register to the memory for using this value outside of a parallel part
* *master*: Executed only by the master thread (the thread which forked off all the others during the execution of the OpenMP directive). No implicit barrier; other team members (threads) not required to reach.

**User-level runtime routines**[[edit](https://en.wikipedia.org/w/index.php?title=OpenMP&action=edit&section=8)]

Used to modify/check the number of threads, detect if the execution context is in a parallel region, how many processors in current system, set/unset locks, timing functions, etc

**Environment variables**[[edit](https://en.wikipedia.org/w/index.php?title=OpenMP&action=edit&section=9)]

A method to alter the execution features of OpenMP applications. Used to control loop iterations scheduling, default number of threads, etc. For example, *OMP\_NUM\_THREADS* is used to specify number of threads for an application.

Implementations[[edit](https://en.wikipedia.org/w/index.php?title=OpenMP&action=edit&section=10)]

OpenMP has been implemented in many commercial compilers. For instance, Visual C++ 2005, 2008, 2010, 2012 and 2013 support it (OpenMP 2.0, in Professional, Team System, Premium and Ultimate editions[[16]](https://en.wikipedia.org/wiki/OpenMP#cite_note-16)[[17]](https://en.wikipedia.org/wiki/OpenMP#cite_note-17)[[18]](https://en.wikipedia.org/wiki/OpenMP#cite_note-18)), as well as [Intel Parallel Studio](https://en.wikipedia.org/wiki/Intel_Parallel_Studio) for various processors.[[19]](https://en.wikipedia.org/wiki/OpenMP#cite_note-19) [Oracle Solaris Studio](https://en.wikipedia.org/wiki/Oracle_Solaris_Studio) compilers and tools support the latest [OpenMP specifications](https://web.archive.org/web/20081004161456/http:/openmp.org/wp/openmp-specifications/) with productivity enhancements for Solaris OS (UltraSPARC and x86/x64) and Linux platforms. The Fortran, C and C++ compilers from [The Portland Group](https://en.wikipedia.org/wiki/The_Portland_Group) also support OpenMP 2.5. [GCC](https://en.wikipedia.org/wiki/GNU_Compiler_Collection) has also supported OpenMP since version 4.2.

Compilers with an implementation of OpenMP 3.0:

* GCC 4.3.1
* Mercurium compiler
* Intel Fortran and C/C++ versions 11.0 and 11.1 compilers, Intel C/C++ and Fortran Composer XE 2011 and Intel Parallel Studio.
* IBM XL compiler[[20]](https://en.wikipedia.org/wiki/OpenMP#cite_note-20)
* Sun Studio 12 update 1 has a full implementation of OpenMP 3.0[[21]](https://en.wikipedia.org/wiki/OpenMP#cite_note-21)

Several compilers support OpenMP 3.1:

* GCC 4.7[[22]](https://en.wikipedia.org/wiki/OpenMP#cite_note-openmp_%E2%80%93_GCC_Wiki-22)
* Intel Fortran and C/C++ compilers 12.1[[23]](https://en.wikipedia.org/wiki/OpenMP#cite_note-23)
* IBM XL C/C++ compilers for AIX and Linux, V13.1[[24]](https://en.wikipedia.org/wiki/OpenMP#cite_note-ibm.com-24) & IBM XL Fortran compilers for AIX and Linux, V14.1[[25]](https://en.wikipedia.org/wiki/OpenMP#cite_note-www-01.ibm.com-25)
* LLVM/Clang 3.7[[26]](https://en.wikipedia.org/wiki/OpenMP#cite_note-Clang_3.7_Release_Notes-26)
* [Absoft Fortran Compilers](https://en.wikipedia.org/wiki/Absoft_Fortran_Compilers) v. 19 for Windows, Mac OS X and Linux[[27]](https://en.wikipedia.org/wiki/OpenMP#cite_note-Absoft_Pro_Fortran_Compilers_and_Debuggers-27)

Compilers supporting OpenMP 4.0:

* GCC 4.9.0 for C/C++, GCC 4.9.1 for Fortran[[22]](https://en.wikipedia.org/wiki/OpenMP#cite_note-openmp_%E2%80%93_GCC_Wiki-22)[[28]](https://en.wikipedia.org/wiki/OpenMP#cite_note-28)
* Intel Fortran and C/C++ compilers 15.0[[29]](https://en.wikipedia.org/wiki/OpenMP#cite_note-29)
* IBM XL C/C++ for Linux, V13.1 (partial)[[24]](https://en.wikipedia.org/wiki/OpenMP#cite_note-ibm.com-24) & XL Fortran for Linux, V15.1 (partial)[[25]](https://en.wikipedia.org/wiki/OpenMP#cite_note-www-01.ibm.com-25)
* LLVM/Clang 3.7 (partial)[[26]](https://en.wikipedia.org/wiki/OpenMP#cite_note-Clang_3.7_Release_Notes-26)

Several Compilers supporting OpenMP 4.5:

* GCC 6 for C/C++ [[30]](https://en.wikipedia.org/wiki/OpenMP#cite_note-30)
* Intel Fortran and C/C++ compilers 17.0, 18.0, 19.0 [[31]](https://en.wikipedia.org/wiki/OpenMP#cite_note-OpenMP_Tools-31)

[Auto-parallelizing](https://en.wikipedia.org/wiki/Automatic_parallelization) compilers that generates source code annotated with OpenMP directives:

* iPat/OMP
* [Parallware](https://en.wikipedia.org/w/index.php?title=Parallware&action=edit&redlink=1)
* PLUTO
* [ROSE (compiler framework)](https://en.wikipedia.org/wiki/ROSE_(compiler_framework))
* S2P by KPIT Cummins Infosystems Ltd.

Several profilers and debuggers expressly support OpenMP:

* [Allinea Distributed Debugging Tool](https://en.wikipedia.org/wiki/Allinea_Distributed_Debugging_Tool) (DDT) – debugger for OpenMP and MPI codes
* [Allinea MAP](https://en.wikipedia.org/wiki/Allinea_MAP) – profiler for OpenMP and MPI codes
* TotalView - debugger from [Rogue Wave Software](https://en.wikipedia.org/wiki/Rogue_Wave_Software) for OpenMP, MPI and serial codes
* ompP – profiler for OpenMP
* VAMPIR – profiler for OpenMP and MPI code

Pros and cons[[edit](https://en.wikipedia.org/w/index.php?title=OpenMP&action=edit&section=11)]

|  |  |
| --- | --- |
|  | This section **needs additional citations for**[**verification**](https://en.wikipedia.org/wiki/Wikipedia:Verifiability). Please help [improve this article](https://en.wikipedia.org/w/index.php?title=OpenMP&action=edit) by [adding citations to reliable sources](https://en.wikipedia.org/wiki/Help:Referencing_for_beginners). Unsourced material may be challenged and removed. *(February 2017) (*[*Learn how and when to remove this template message*](https://en.wikipedia.org/wiki/Help:Maintenance_template_removal)*)* |

Pros:

* Portable multithreading code (in C/C++ and other languages, one typically has to call platform-specific primitives in order to get multithreading).
* Simple: need not deal with message passing as [MPI](https://en.wikipedia.org/wiki/Message_Passing_Interface) does.
* Data layout and decomposition is handled automatically by directives.
* Scalability comparable to [MPI](https://en.wikipedia.org/wiki/Message_Passing_Interface) on shared-memory systems.[[32]](https://en.wikipedia.org/wiki/OpenMP#cite_note-ReferenceA-32)
* Incremental parallelism: can work on one part of the program at one time, no dramatic change to code is needed.
* Unified code for both serial and parallel applications: OpenMP constructs are treated as comments when sequential compilers are used.
* Original (serial) code statements need not, in general, be modified when parallelized with OpenMP. This reduces the chance of inadvertently introducing bugs.
* Both [coarse-grained](https://en.wikipedia.org/wiki/Granularity_(parallel_computing)) and [fine-grained](https://en.wikipedia.org/wiki/Granularity_(parallel_computing)) parallelism are possible.
* In irregular multi-physics applications which do not adhere solely to the [SPMD](https://en.wikipedia.org/wiki/SPMD) mode of computation, as encountered in tightly coupled fluid-particulate systems, the flexibility of OpenMP can have a big performance advantage over [MPI](https://en.wikipedia.org/wiki/Message_Passing_Interface).[[32]](https://en.wikipedia.org/wiki/OpenMP#cite_note-ReferenceA-32)[[33]](https://en.wikipedia.org/wiki/OpenMP#cite_note-33)
* Can be used on various accelerators such as [GPGPU](https://en.wikipedia.org/wiki/GPGPU)[[34]](https://en.wikipedia.org/wiki/OpenMP#cite_note-34) and [FPGAs](https://en.wikipedia.org/wiki/Field-programmable_gate_array).

Cons:

* Risk of introducing difficult to debug synchronization bugs and [race conditions](https://en.wikipedia.org/wiki/Race_condition).[[35]](https://en.wikipedia.org/wiki/OpenMP#cite_note-35)[[36]](https://en.wikipedia.org/wiki/OpenMP#cite_note-36)
* As of 2017 only runs efficiently in shared-memory multiprocessor platforms (see however Intel's [Cluster OpenMP](http://software.intel.com/en-us/articles/cluster-openmp-for-intel-compilers) and other [distributed shared memory](https://en.wikipedia.org/wiki/Distributed_shared_memory) platforms).
* Requires a compiler that supports OpenMP.
* Scalability is limited by memory architecture.
* No support for [compare-and-swap](https://en.wikipedia.org/wiki/Compare-and-swap).[[37]](https://en.wikipedia.org/wiki/OpenMP#cite_note-37)
* Reliable error handling is missing.
* Lacks fine-grained mechanisms to control thread-processor mapping.
* High chance of accidentally writing [false sharing](https://en.wikipedia.org/wiki/False_sharing) code.

Performance expectations[[edit](https://en.wikipedia.org/w/index.php?title=OpenMP&action=edit&section=12)]

One might expect to get an *N* times [speedup](https://en.wikipedia.org/wiki/Speedup) when running a program parallelized using OpenMP on a *N* processor platform. However, this seldom occurs for these reasons:

* When a dependency exists, a process must wait until the data it depends on is computed.
* When multiple processes share a non-parallel proof resource (like a file to write in), their requests are executed sequentially. Therefore, each thread must wait until the other thread releases the resource.
* A large part of the program may not be parallelized by OpenMP, which means that the theoretical upper limit of speedup is limited according to [Amdahl's law](https://en.wikipedia.org/wiki/Amdahl%27s_law).
* N processors in a [symmetric multiprocessing](https://en.wikipedia.org/wiki/Symmetric_multiprocessing) (SMP) may have N times the computation power, but the [memory bandwidth](https://en.wikipedia.org/wiki/Memory_bandwidth) usually does not scale up N times. Quite often, the original memory path is shared by multiple processors and performance degradation may be observed when they compete for the shared memory bandwidth.
* Many other common problems affecting the final speedup in parallel computing also apply to OpenMP, like [load balancing](https://en.wikipedia.org/wiki/Load_balancing_(computing)) and synchronization overhead.
* Compiler optimisation may not be as effective when invoking OpenMP. This can commonly lead to a single-threaded OpenMP program running slower than the same code compiled without an OpenMP flag (which will be fully serial).

Thread affinity[[edit](https://en.wikipedia.org/w/index.php?title=OpenMP&action=edit&section=13)]

Some vendors recommend setting the [processor affinity](https://en.wikipedia.org/wiki/Processor_affinity) on OpenMP threads to associate them with particular processor cores.[[38]](https://en.wikipedia.org/wiki/OpenMP#cite_note-38)[[39]](https://en.wikipedia.org/wiki/OpenMP#cite_note-39)[[40]](https://en.wikipedia.org/wiki/OpenMP#cite_note-40) This minimizes thread migration and context-switching cost among cores. It also improves the data locality and reduces the cache-coherency traffic among the cores (or processors).

Benchmarks[[edit](https://en.wikipedia.org/w/index.php?title=OpenMP&action=edit&section=14)]

A variety of benchmarks has been developed to demonstrate the use of OpenMP, test its performance and evaluate correctness.

Simple examples

* [OmpSCR: OpenMP Source Code Repository](https://sourceforge.net/projects/ompscr/)

Performance benchmarks include:

* [EPCC OpenMP/MPI micro-benchmark suite](https://www.epcc.ed.ac.uk/research/computing/performance-characterisation-and-benchmarking/epcc-openmpmpi-micro-benchmark)
* [NAS Parallel Benchmark](http://aces.snu.ac.kr/software/snu-npb/)
* [Barcelona OpenMP Task Suite](https://github.com/bsc-pm/bots) a collection of applications that allow to test OpenMP tasking implementations.
* SPEC series
  + [SPEC OMP 2012](https://www.spec.org/omp2012/)
  + [The SPEC ACCEL benchmark suite](https://www.spec.org/accel/) testing OpenMP 4 target offloading API
  + [The SPEChpc® 2002 benchmark](https://www.spec.org/hpc2002/)
* [ASC Sequoia Benchmark Codes](https://asc.llnl.gov/sequoia/benchmarks/)
* [CORAL benchmarks](https://asc.llnl.gov/CORAL-benchmarks/)
* [Rodinia](https://rodinia.cs.virginia.edu/doku.php) focusing on accelerators.
* [Problem Based Benchmark Suite](http://www.cs.cmu.edu/~pbbs/benchmarks.html)

Correctness benchmarks include:

* [OpenMP Validation Suite](https://github.com/uhhpctools/omp-validation)
* [OpenMP Validation and Verification Testsuite](https://crpl.cis.udel.edu/ompvvsollve/)
* [DataRaceBench](https://github.com/LLNL/dataracebench) is a benchmark suite designed to systematically and quantitatively evaluate the effectiveness of OpenMP data race detection tools.
* [AutoParBench](https://github.com/LLNL/AutoParBench) is a benchmark suite to evaluate compilers and tools which can automatically insert OpenMP directives.

See also[[edit](https://en.wikipedia.org/w/index.php?title=OpenMP&action=edit&section=15)]

|  |  |
| --- | --- |
|  | This "[see also](https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Layout#%22See_also%22_section)" section **may contain an excessive number of suggestions**. Please ensure that only the most relevant links are given, that they are not [red links](https://en.wikipedia.org/wiki/Wikipedia:Red_link), and that any links are not already in this article. *(February 2017) (*[*Learn how and when to remove this template message*](https://en.wikipedia.org/wiki/Help:Maintenance_template_removal)*)* |

* [Chapel (programming language)](https://en.wikipedia.org/wiki/Chapel_(programming_language))
* [Cilk](https://en.wikipedia.org/wiki/Cilk)
* [Cilk Plus](https://en.wikipedia.org/wiki/Cilk_Plus)
* [Message Passing Interface](https://en.wikipedia.org/wiki/Message_Passing_Interface)
* [Concurrency (computer science)](https://en.wikipedia.org/wiki/Concurrency_(computer_science))
* [Heterogeneous System Architecture](https://en.wikipedia.org/wiki/Heterogeneous_System_Architecture)
* [Parallel computing](https://en.wikipedia.org/wiki/Parallel_computing)
* [Parallel programming model](https://en.wikipedia.org/wiki/Parallel_programming_model)
* [POSIX Threads](https://en.wikipedia.org/wiki/POSIX_Threads)
* [Unified Parallel C](https://en.wikipedia.org/wiki/Unified_Parallel_C)
* [X10 (programming language)](https://en.wikipedia.org/wiki/X10_(programming_language))
* [Parallel Virtual Machine](https://en.wikipedia.org/wiki/Parallel_Virtual_Machine)
* [Bulk synchronous parallel](https://en.wikipedia.org/wiki/Bulk_synchronous_parallel)
* [Grand Central Dispatch](https://en.wikipedia.org/wiki/Grand_Central_Dispatch)
* [Partitioned global address space](https://en.wikipedia.org/wiki/Partitioned_global_address_space)
* [GPGPU](https://en.wikipedia.org/wiki/GPGPU)
* [CUDA](https://en.wikipedia.org/wiki/CUDA) – Nvidia
* [Octopiler](https://en.wikipedia.org/wiki/Octopiler)
* [OpenCL](https://en.wikipedia.org/wiki/OpenCL)
* [OpenACC](https://en.wikipedia.org/wiki/OpenACC)
* [SequenceL](https://en.wikipedia.org/wiki/SequenceL)
* [Enduro/X](https://en.wikipedia.org/wiki/Enduro/X)

References[[edit](https://en.wikipedia.org/w/index.php?title=OpenMP&action=edit&section=16)]

* 1. ^ [Jump up to:***a***](https://en.wikipedia.org/wiki/OpenMP#cite_ref-Board_1-0) [***b***](https://en.wikipedia.org/wiki/OpenMP#cite_ref-Board_1-1) [***c***](https://en.wikipedia.org/wiki/OpenMP#cite_ref-Board_1-2) [*"About the OpenMP ARB and"*](https://web.archive.org/web/20130809153922/http:/openmp.org/wp/about-openmp/)*. OpenMP.org. 2013-07-11. Archived from*[*the original*](http://openmp.org/wp/about-openmp/)*on 2013-08-09. Retrieved 2013-08-14.*
  2. ^ [Jump up to:***a***](https://en.wikipedia.org/wiki/OpenMP#cite_ref-openmp.org_2-0) [***b***](https://en.wikipedia.org/wiki/OpenMP#cite_ref-openmp.org_2-1) [*"OpenMP Compilers & Tools"*](https://www.openmp.org/resources/openmp-compilers-tools/)*. OpenMP.org. November 2019. Retrieved 2020-03-05.*
  3. ^ [Jump up to:***a***](https://en.wikipedia.org/wiki/OpenMP#cite_ref-OSConcepts_3-0) [***b***](https://en.wikipedia.org/wiki/OpenMP#cite_ref-OSConcepts_3-1) *Gagne, Abraham Silberschatz, Peter Baer Galvin, Greg (2012-12-17). Operating system concepts (9th ed.). Hoboken, N.J.: Wiley. pp. 181–182.*[*ISBN*](https://en.wikipedia.org/wiki/ISBN_(identifier))[*978-1-118-06333-0*](https://en.wikipedia.org/wiki/Special:BookSources/978-1-118-06333-0)*.*
  4. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-4) [OpenMP Tutorial at Supercomputing 2008](http://openmp.org/wp/2008/10/openmp-tutorial-at-supercomputing-2008/)
  5. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-5) [Using OpenMP – Portable Shared Memory Parallel Programming – Download Book Examples and Discuss](http://openmp.org/wp/2009/04/download-book-examples-and-discuss/)
  6. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-6) *Costa, J.J.; et al. (May 2006). "Running OpenMP applications efficiently on an everything-shared SDSM". Journal of Parallel and Distributed Computing.****66****(5): 647–658.*[*doi*](https://en.wikipedia.org/wiki/Doi_(identifier))*:*[*10.1016/j.jpdc.2005.06.018*](https://doi.org/10.1016%2Fj.jpdc.2005.06.018)*.*
  7. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-7) *Basumallik, Ayon; Min, Seung-Jai; Eigenmann, Rudolf (2007). Programming Distributed Memory Sytems [sic] using OpenMP. Proceedings of the 2007 IEEE International Parallel and Distributed Processing Symposium. New York: IEEE Press. pp. 1–8.*[*CiteSeerX*](https://en.wikipedia.org/wiki/CiteSeerX_(identifier))[*10.1.1.421.8570*](https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.421.8570)*.*[*doi*](https://en.wikipedia.org/wiki/Doi_(identifier))*:*[*10.1109/IPDPS.2007.370397*](https://doi.org/10.1109%2FIPDPS.2007.370397)*.*[*ISBN*](https://en.wikipedia.org/wiki/ISBN_(identifier))[*978-1-4244-0909-9*](https://en.wikipedia.org/wiki/Special:BookSources/978-1-4244-0909-9)*.* A [preprint is available on Chen Ding's home page](https://www.cs.rochester.edu/~cding/Announcements/HIPS07/openmp.pdf); see especially Section 3 on Translation of OpenMP to MPI.
  8. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-8) *Wang, Jue; Hu, ChangJun; Zhang, JiLin; Li, JianJiang (May 2010).*[*"OpenMP compiler for distributed memory architectures"*](https://doi.org/10.1007/s11432-010-0074-0)*. Science China Information Sciences.****53****(5): 932–944.*[*doi*](https://en.wikipedia.org/wiki/Doi_(identifier))*:*[*10.1007/s11432-010-0074-0*](https://doi.org/10.1007%2Fs11432-010-0074-0)*.* (As of 2016 the KLCoMP software described in this paper does not appear to be publicly available)
  9. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-9) [Cluster OpenMP](https://software.intel.com/en-us/articles/cluster-openmp-for-intel-compilers) (a product that used to be available for [Intel C++ Compiler](https://en.wikipedia.org/wiki/Intel_C%2B%2B_Compiler) versions 9.1 to 11.1 but was dropped in 13.0)
  10. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-10) *Ayguade, Eduard; Copty, Nawal; Duran, Alejandro; Hoeflinger, Jay; Lin, Yuan; Massaioli, Federico; Su, Ernesto; Unnikrishnan, Priya; Zhang, Guansong (2007).*[*A proposal for task parallelism in OpenMP*](http://people.ac.upc.edu/aduran/papers/2007/tasks_iwomp07.pdf)*(PDF). Proc. Int'l Workshop on OpenMP.*
  11. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-11) [*"OpenMP Application Program Interface, Version 3.0"*](http://www.openmp.org/mp-documents/spec30.pdf)*(PDF). openmp.org. May 2008. Retrieved 2014-02-06.*
  12. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-12) *LaGrone, James; Aribuki, Ayodunni; Addison, Cody; Chapman, Barbara (2011). A Runtime Implementation of OpenMP Tasks. Proc. Int'l Workshop on OpenMP. pp. 165–178.*[*CiteSeerX*](https://en.wikipedia.org/wiki/CiteSeerX_(identifier))[*10.1.1.221.2775*](https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.221.2775)*.*[*doi*](https://en.wikipedia.org/wiki/Doi_(identifier))*:*[*10.1007/978-3-642-21487-5\_13*](https://doi.org/10.1007%2F978-3-642-21487-5_13)*.*
  13. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-13) [*"OpenMP 4.0 API Released"*](https://web.archive.org/web/20131109175921/http:/openmp.org/wp/openmp-40-api-released/)*. OpenMP.org. 2013-07-26. Archived from*[*the original*](http://openmp.org/wp/openmp-40-api-released/)*on 2013-11-09. Retrieved 2013-08-14.*
  14. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-14) [*"OpenMP Application Program Interface, Version 4.0"*](http://www.openmp.org/mp-documents/OpenMP4.0.0.pdf)*(PDF). openmp.org. July 2013. Retrieved 2014-02-06.*
  15. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-15) [*"Tutorial – Parallel for Loops with OpenMP"*](http://supercomputingblog.com/openmp/tutorial-parallel-for-loops-with-openmp/)*. 2009-07-14.*
  16. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-16) [Visual C++ Editions, Visual Studio 2005](http://msdn2.microsoft.com/en-us/library/hs24szh9(vs.80).aspx)
  17. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-17) [Visual C++ Editions, Visual Studio 2008](http://msdn2.microsoft.com/en-us/library/hs24szh9(vs.90).aspx)
  18. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-18) [Visual C++ Editions, Visual Studio 2010](http://msdn2.microsoft.com/en-us/library/hs24szh9(vs.100).aspx)
  19. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-19) David Worthington, ["Intel addresses development life cycle with Parallel Studio"](http://www.sdtimes.com/intel_addresses_development_life_cycle_with_parallel_studio/about_intel_and_multicore/33497) [Archived](https://web.archive.org/web/20120215032407/http:/www.sdtimes.com/INTEL_ADDRESSES_DEVELOPMENT_LIFE_CYCLE_WITH_PARALLEL_STUDIO/About_INTEL_and_MULTICORE/33497) 2012-02-15 at the [Wayback Machine](https://en.wikipedia.org/wiki/Wayback_Machine), SDTimes, 26 May 2009 (accessed 28 May 2009)
  20. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-20) ["XL C/C++ for Linux Features"](http://www-01.ibm.com/software/awdtools/xlcpp/linux/features/?S_CMP=rnav), (accessed 9 June 2009)
  21. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-21) [*"Oracle Technology Network for Java Developers | Oracle Technology Network | Oracle"*](http://developers.sun.com/sunstudio/features/)*. Developers.sun.com. Retrieved 2013-08-14.*
  22. ^ [Jump up to:***a***](https://en.wikipedia.org/wiki/OpenMP#cite_ref-openmp_%E2%80%93_GCC_Wiki_22-0) [***b***](https://en.wikipedia.org/wiki/OpenMP#cite_ref-openmp_%E2%80%93_GCC_Wiki_22-1) [*"openmp – GCC Wiki"*](https://gcc.gnu.org/wiki/openmp)*. Gcc.gnu.org. 2013-07-30. Retrieved 2013-08-14.*
  23. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-23) *Submitted by Patrick Kennedy... on Fri, 09/02/2011 – 11:28 (2011-09-06).*[*"Intel® C++ and Fortran Compilers now support the OpenMP\* 3.1 Specification | Intel® Developer Zone"*](http://software.intel.com/en-us/articles/intel-c-and-fortran-compilers-now-support-the-openmp-31-specification/)*. Software.intel.com. Retrieved 2013-08-14.*
  24. ^ [Jump up to:***a***](https://en.wikipedia.org/wiki/OpenMP#cite_ref-ibm.com_24-0) [***b***](https://en.wikipedia.org/wiki/OpenMP#cite_ref-ibm.com_24-1) <https://www.ibm.com/support/docview.wss?uid=swg27007322&aid=1>
  25. ^ [Jump up to:***a***](https://en.wikipedia.org/wiki/OpenMP#cite_ref-www-01.ibm.com_25-0) [***b***](https://en.wikipedia.org/wiki/OpenMP#cite_ref-www-01.ibm.com_25-1) <http://www-01.ibm.com/support/docview.wss?uid=swg27007323&aid=1>
  26. ^ [Jump up to:***a***](https://en.wikipedia.org/wiki/OpenMP#cite_ref-Clang_3.7_Release_Notes_26-0) [***b***](https://en.wikipedia.org/wiki/OpenMP#cite_ref-Clang_3.7_Release_Notes_26-1) [*"Clang 3.7 Release Notes"*](http://llvm.org/releases/3.7.0/tools/clang/docs/ReleaseNotes.html#openmp-support)*. llvm.org. Retrieved 2015-10-10.*
  27. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-Absoft_Pro_Fortran_Compilers_and_Debuggers_27-0) [*"Absoft Home Page"*](https://www.absoft.com/)*. Retrieved 2019-02-12.*
  28. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-28) [*"GCC 4.9 Release Series – Changes"*](https://www.gnu.org/software/gcc/gcc-4.9/changes.html)*. www.gnu.org.*
  29. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-29) [*"OpenMP\* 4.0 Features in Intel Compiler 15.0"*](https://software.intel.com/en-us/articles/openmp-40-features-in-intel-compiler-150)*. Software.intel.com. 2014-08-13.*
  30. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-30) [*"GCC 6 Release Series - Changes"*](https://www.gnu.org/software/gcc/gcc-6/changes.html)*. www.gnu.org.*
  31. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-OpenMP_Tools_31-0) [*"OpenMP Compilers & Tools"*](https://www.openmp.org/resources/openmp-compilers-tools/)*. openmp.org. www.openmp.org. Retrieved 29 October 2019.*
  32. ^ [Jump up to:***a***](https://en.wikipedia.org/wiki/OpenMP#cite_ref-ReferenceA_32-0) [***b***](https://en.wikipedia.org/wiki/OpenMP#cite_ref-ReferenceA_32-1) *Amritkar, Amit; Tafti, Danesh; Liu, Rui; Kufrin, Rick; Chapman, Barbara (2012). "OpenMP parallelism for fluid and fluid-particulate systems". Parallel Computing.****38****(9): 501.*[*doi*](https://en.wikipedia.org/wiki/Doi_(identifier))*:*[*10.1016/j.parco.2012.05.005*](https://doi.org/10.1016%2Fj.parco.2012.05.005)*.*
  33. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-33) *Amritkar, Amit; Deb, Surya; Tafti, Danesh (2014).*[*"Efficient parallel CFD-DEM simulations using OpenMP"*](https://en.wikipedia.org/wiki/CFD-DEM)*. Journal of Computational Physics.****256****: 501.*[*Bibcode*](https://en.wikipedia.org/wiki/Bibcode_(identifier))*:*[*2014JCoPh.256..501A*](https://ui.adsabs.harvard.edu/abs/2014JCoPh.256..501A)*.*[*doi*](https://en.wikipedia.org/wiki/Doi_(identifier))*:*[*10.1016/j.jcp.2013.09.007*](https://doi.org/10.1016%2Fj.jcp.2013.09.007)*.*
  34. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-34) [OpenMP Accelerator Support for GPUs](https://www.openmp.org/updates/openmp-accelerator-support-gpus/)
  35. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-35) [Detecting and Avoiding OpenMP Race Conditions in C++](http://developers.sun.com/solaris/articles/cpp_race.html)
  36. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-36) [Alexey Kolosov, Evgeniy Ryzhkov, Andrey Karpov 32 OpenMP traps for C++ developers](http://software.intel.com/en-us/articles/32-openmp-traps-for-c-developers)
  37. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-37) Stephen Blair-Chappell, Intel Corporation, Becoming a Parallel Programming Expert in Nine Minutes, presentation on [ACCU](https://en.wikipedia.org/wiki/ACCU_(organisation)) 2010 conference
  38. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-38) *Chen, Yurong (2007-11-15). "Multi-Core Software". Intel Technology Journal.****11****(4).*[*doi*](https://en.wikipedia.org/wiki/Doi_(identifier))*:*[*10.1535/itj.1104.08*](https://doi.org/10.1535%2Fitj.1104.08)*.*
  39. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-39) [*"OMPM2001 Result"*](http://www.spec.org/omp/results/res2008q1/omp2001-20080128-00288.html)*. SPEC. 2008-01-28.*
  40. [**^**](https://en.wikipedia.org/wiki/OpenMP#cite_ref-40) [*"OMPM2001 Result"*](http://www.spec.org/omp/results/res2003q2/omp2001-20030401-00079.html)*. SPEC. 2003-04-01.*

Further reading[[edit](https://en.wikipedia.org/w/index.php?title=OpenMP&action=edit&section=17)]

* Quinn Michael J, *Parallel Programming in C with MPI and OpenMP* McGraw-Hill Inc. 2004. [ISBN](https://en.wikipedia.org/wiki/ISBN_(identifier)) [0-07-058201-7](https://en.wikipedia.org/wiki/Special:BookSources/0-07-058201-7)
* R. Chandra, R. Menon, L. Dagum, D. Kohr, D. Maydan, J. McDonald, *Parallel Programming in OpenMP.* Morgan Kaufmann, 2000. [ISBN](https://en.wikipedia.org/wiki/ISBN_(identifier)) [1-55860-671-8](https://en.wikipedia.org/wiki/Special:BookSources/1-55860-671-8)
* R. Eigenmann (Editor), M. Voss (Editor), *OpenMP Shared Memory Parallel Programming: International Workshop on OpenMP Applications and Tools, WOMPAT 2001, West Lafayette, IN, USA, July 30–31, 2001.* (Lecture Notes in Computer Science). Springer 2001. [ISBN](https://en.wikipedia.org/wiki/ISBN_(identifier)) [3-540-42346-X](https://en.wikipedia.org/wiki/Special:BookSources/3-540-42346-X)
* B. Chapman, G. Jost, R. van der Pas, D.J. Kuck (foreword), *Using OpenMP: Portable Shared Memory Parallel Programming.* The MIT Press (October 31, 2007). [ISBN](https://en.wikipedia.org/wiki/ISBN_(identifier)) [0-262-53302-2](https://en.wikipedia.org/wiki/Special:BookSources/0-262-53302-2)
* Parallel Processing via MPI & OpenMP, M. Firuziaan, O. Nommensen. Linux Enterprise, 10/2002
* [MSDN Magazine article on OpenMP](https://web.archive.org/web/20080705180752/http:/msdn.microsoft.com/msdnmag/issues/05/10/OpenMP/default.aspx)
* [SC08 OpenMP Tutorial](http://openmp.org/mp-documents/omp-hands-on-SC08.pdf) (PDF) – Hands-On Introduction to OpenMP, Mattson and Meadows, from SC08 (Austin)
* [OpenMP Specifications](https://www.openmp.org/specifications/)
* [Parallel Programming in Fortran 95 using OpenMP](http://www.openmp.org/wp-content/uploads/F95_OpenMPv1_v2.pdf) (PDF)

External links[[edit](https://en.wikipedia.org/w/index.php?title=OpenMP&action=edit&section=18)]

* [Official website](https://www.openmp.org/), includes the latest OpenMP specifications, links to resources, lively set of forums where questions can be asked and are answered by OpenMP experts and implementors
* [OpenMPCon](https://openmpcon.org/), website of the OpenMP Developers Conference
* [IWOMP](http://www.iwomp.org/), website for the annual International Workshop on OpenMP
* [UK OpenMP Users](https://ukopenmpusers.co.uk/), website for the UK OpenMP Users group and conference
* [IBM Octopiler](http://domino.research.ibm.com/comm/research_projects.nsf/pages/cellcompiler.index.html) with OpenMP support
* [Blaise Barney, Lawrence Livermore National Laboratory site on OpenMP](https://computing.llnl.gov/tutorials/openMP/)
* [Combining OpenMP and MPI](https://web.archive.org/web/20140223093700/http:/geco.mines.edu/workshop/aug2010/slides/thu/hybrid.pdf) (PDF)
* [Mixing MPI and OpenMP](http://www.slac.stanford.edu/comp/unix/farm/mpi_and_openmp.html)
* [Measure and visualize OpenMP parallelism](https://myarm.com/blog-measure-and-visualize-parallelism.html) by means of a [C++](https://en.wikipedia.org/wiki/C%2B%2B) routing planner calculating the [Speedup](https://en.wikipedia.org/wiki/Speedup) factor