GCC/OpenMP Update

Tobias Burnus
Catherine Moore

SIEMENS

Agenda

- GCC Overview
  - GCC Community
  - GCC Release Cycle
  - GCC 12 OpenMP Support
  - Testing OpenMP @Siemens
  - GCC Resources
- OpenMP 5.0 + 5.1 Support
- Specification Corner-Cases
- Command-Line Options & Tricks
- Conclusion

OpenMP Community in GCC

- **Welcoming to new contributors**
  - Developer Certificate of Origin (DCO) or FSF copyright needed
- **Siemens (funded by ORNL and the DOE)**
  - Six active developers, led by Tobias Burnus, working on OpenMP functionality and performance
  - Major contributor; most of Fortran development
- **OpenMP Patch Review**
  - Maintainer - Jakub Jelinek
  - Reviews and contributes patches
- **Others**
  - Participation on ad-hoc basis

SIEMENS

GCC Release Cycle

Date	Release	Development Branch
April 2021	GCC 11	OG11 Branch
Spring 2022	GCC 12	OG12 Branch
Spring 2023	GCC 13	OG13 Branch

OG Development Branches

- GCC GIT branch devel/omp/gcc-11 etc.
  - Maintained by Siemens developers
- Offers early access to OpenMP offloading features not in the official release
- Allows development to continue during GCC pre-release quiet periods
- Recommended for use for latest performance and functionality
- GCC Open Development: Usually May through November

GCC 12 OpenMP Support

OpenMP Revision	Support Level	NVIDIA Offloading	AMD Offloading
OpenMP 4.5	Fully Supported	Yes	Yes
OpenMP 5.0	Partial Support	Yes	Yes
OpenMP 5.1	Minimal Support	Yes	Yes

AMD Offloading Support for MI100 is complete; Newer AMD offerings planned for 2022

Testing OpenMP @Siemens

Nightly and weekly tests targeting AMD M100 and NVIDIA Volta

Test suites

- sollve_vv
- omptest
- OvO
- Babelstream

- SPEC ACCEL
- SPEChpc 2021

- GCC's DejaGNU test suite
  - C/C++
  - gfortran
  - libgomp

SIEMENS

GCC Resources

GCC Offloading: https://gcc.gnu.org/wiki/Offloading

- Building GCC for offloading
- Options for building applications for offloading

OpenMP implementation status (development branch):

- https://gcc.gnu.org/onlinedocs/libgomp/OpenMP-Implementation-Status.html

Libgomp manual:

- GCC 11.2 (stable): https://gcc.gnu.org/onlinedocs/gcc-11.2.0/libgomp/
- Development branch: https://gcc.gnu.org/onlinedocs/libgomp/

General help with GCC

- mailing list: https://gcc.gnu.org/pipermail/gcc-help/

SIEMENS

Agenda

- GCC Overview
- OpenMP 5.0 + 5.1 Support
  - OpenMP Support in GCC
  - OpenMP 5.0 + 5.1 Features Supported in GCC 12
  - OpenMP 5.0 + 5.1 Features Unsupported, Planed for GCC 13
- Specification Corner-Cases
- Command-Line Options & Tricks

OpenMP Support in GCC

- GCC: Compiler for C, C++, Fortran, Ada, D, go, …
  - C17 (steps to C2x), C++20 (steps to C++23)
  - Fortran 2008 + coarray + interop TS, initial F2018
- OpenMP and OpenACC 2.6 with C/C++/Fortran
- Supported archs: aarch64, alpha, arc, arm, avr, bfin, …
- GCC offloading-support packages of Linux distributions
  - Debian/Ubuntu: gcc-11-offload-{nvptx,amdgcn}
  - (open)SUSE: cross-{nvptx,amdgcn}-gcc11
  - Red Hat/Fedora: {gcc,libgomp}-offload-nvptx (currently no amdgcn)

New features

- GCC 9 (2019): OpenMP 4.5 (C/C++, Fortran mostly), some 5.0, OpenACC 2.5
- GCC 10 (2020): More of OpenMP 5.0, OpenACC 2.6
- GCC 11 (2021): More of OpenMP 5.0
- GCC 12 (2022): Some OpenMP 5.1, more 5.0 (esp. Fortran)

Supported Releases

GCC 11.2 (changes)
Status: 2021-07-28 (regression fixes & docs only).
Serious regressions. All regressions.

GCC 10.3 (changes)
Status: 2021-04-08 (regression fixes & docs only).
Serious regressions. All regressions.

GCC 9.4 (changes)
Status: 2021-06-01 (regression fixes & docs only).
Serious regressions. All regressions.

Development: GCC 12.0 (release criteria, changes)
Status: 2022-01-17 (regression fixes & docs only).
Serious regressions. All regressions.

SIEMENS

OpenMP 5.0 Features Supported in GCC 12

Iterators	**omp_fulfill_event** runtime routine
target-offload-var ICV and OMP_TARGET_OFFLOAD env variable	**reduction** and **in_reduction** clauses on **taskloop** and **taskloop simd** constructs
Nested-parallel changes to *max-active-levels-var* ICV	**taskloop** construct cancelable by **cancel** construct
teams construct outside an enclosing target region	**mutexinouset** *dependence-type* for **depend** clause
!= as relational-op in canonical loop form for C/C++	Predefined memory spaces, memory allocators, allocator traits
nonmonotonic as default loop schedule modifier for worksharing-loop constructs	Memory management routines
Clauses **if**, **nontemporal** and **order(concurrent)** in **simd** construct	**use_device_addr** clause on target data
atomic constructs in **simd**	Implicit **declare target** directive
loop construct	C/C++'s lvalue expressions in depend clauses
order(concurrent) clause	Nested **declare target** directive
scan directive and **in_scan** modifier for the **reduction** clause	Combined **master** constructs
in_reduction clause on **task** constructs	**depend** clause on taskwait
task_reduction clause with **taskgroup**	Weak memory ordering clauses on **atomic** and **flush** construct
task modifier to **reduction** clause	**depobj** construct and depend objects
affinity clause to **task** construct	Lock hints were renamed to synchronization hints
detach clause to **task** construct	**conditional** modifier to **lastprivate** clause
close *map-type-modifier*	**defaultmap** extensions
omp_get_supported_active_levels routine	Runtime routines and environment variables to display runtime thread affinity information
omp_pause_resource and **omp_pause_resource_all** runtime routines	**omp_get_device_num** runtime routine
Supporting C++'s range-based for loop	

SIEMENS

OpenMP 5.1 Features Supported in GCC 12

OpenMP directive as C++ attribute specifiers
nothing directive
error directive
masked construct
scope directive
strict modifier in the **grainsize** and **num_tasks** clauses of the **taskloop** construct
thread_limit clause to **target** construct
Extensions to the **atomic** directive
seq_cst clause on a **flush** construct
private and **firstprivate** argument to **default** clause in C and C++
omp_set_num_teams, **omp_set_teams_thread_limit** runtime routines
omp_get_max_teams, **omp_get_teams_thread_limit** runtime routines
omp_calloc, **omp_realloc** runtime routines
omp_aligned_alloc and **omp_aligned_calloc** runtime routines
omp_alloctrait_key_t enum: **omp_atv_serialized** added, **omp_atv_default** changed
OMP_PLACES syntax extensions
OMP_NUM_TEAMS and OMP_TEAMS_THREAD_LIMIT environment variables
Support of strictly structured blocks in Fortran
Support of structured block sequences in C/C++
unconstrained and **reproducible** modifiers on order clause
omp_display_env runtime routine

OpenMP 5.0 Features Unsupported in GCC 12

Feature	Development Branch Support	Planned for GCC 13
Array shaping	No	**Yes**
Array sections with non-unit strides in C and C++	No	**Yes**
metadirective directive	**Yes**	**Yes**
Collapse of associated loops that are imperfectly nested loops	No	**Yes**
allocate directive	**Yes**	**Yes**
Discontiguous array section with **target update** construct	No	**Yes**
C/C++'s lvalue expressions in **to**, **from**, and **map** clauses	No	**Yes**
declare mapper directive	No	**Yes**
OMPT interface	No	No
OMPD interface	No	No

OpenMP 5.1 Features Planned for GCC 13 – and beyond

omp_all_memory reserved locator	
target_device trait in OpenMP Context	**omp_target_memcpy_async** runtime routine
target_device selector set in context selectors	**omp_target_memcpy_rect_async** runtime routine
C/C++'s **declare variant** directive: elision support of preprocessed code	**omp_get_mapped_ptr** runtime routine
declare variant: new clauses **adjust_args** and **append_args**	*ompt_scope_endpoint_t enum: ompt_scope_beginend*
dispatch construct	*ompt_sync_region_t enum additions*
device-specific ICV settings the environment variables	*ompt_state_t enum: ompt_state_wait_barrier_implementation*
assume directive	*ompt_state_t enum: ompt_state_wait_barrier_teams*
Loop transformation constructs	*ompt_callback_target_data_op_emi_t, ompt_callback_target_emi_t*
has_device_addr clause to target construct	*ompt_callback_target_map_emi_t and ompt_callback_target_submit_emi_t*
iterators in **target update** motion clauses and **map** clauses	*ompt_callback_error_t type*
indirect calls to the device version of a procedure or function in target regions	**nowait** clause in taskwait directive
inoutset argument to the **depend** clause	**interop** directive
present argument to **defaultmap** clause	**omp_interop_t** object support in runtime routines
omp_target_is_accessible runtime routine	

SIEMENS

OpenMP 5.0 Features Partially Supported in GCC 12
declare variant directive – *simd traits not handled correctly*
requires directive – o*nly fulfillable requirement are* ***atomic_default_mem_order*** *and* ***dynamic_allocators***
Non-rectangular loop nests – *Fortran support missing, planned for GCC 13*
in_reduction clause on target constructs – ***nowait*** *is only a stub*
ancestor modifier on device clause – *reverse offload unsupported*
Map-order clarifications
Mapping C/C++ pointer variables and to assign the address of device memory mapped by an array section
Mapping of Fortran pointer and allocatable variables, including pointer and allocatable components of variables – *Mapping of vars with allocatable components unspported, planned for GCC 13*

OpenMP 5.1 Features Partially Supported in GCC 12
align clause/modifier in **allocate** directive/clause and **allocator** directive — *Fortran support missing, planned for GCC 13*

Planned for GCC 13:

- Unified shared memory support with NVIDIA GPUs

SIEMENS

Agenda

- GCC Overview
- OpenMP 5.0 + 5.1 Support
- Specification Corner Cases
  - OpenMP Specification
  - Example for a Minor Spec Issue
  - OMP_TARGET_OFFLOAD=mandatory
  - Spec Work from Our Side
- Command-Line Options & Tricks
- Conclusion

OpenMP Specification

Roughly annual releases

- Either a new OpenMP release or a Technical Report (TR) as snapshot/preview of the next release
- Dot releases have minor changes
- Main change in 5.2: reorganization and syntax representation

Stakeholders

- OpenMP Architecture Review Board (ARB) has 33 members representing GCC: SIEMENS, SUSE, Red Hat (via IBM)
- Weekly language spec meeting (plus subcommittees meetings)

Issues keep popping up: unclear/underspecified, oversights, missing updates after changes/extensions elsewhere

- Despite: text-change discussions, two-step voting, pre-merge proof reading, and whole-document proof reading

OpenMP Specification – Example of a Minor Spec Issue

Fortran – Optional *End-Directive* and *Strictly Structured Blocks*

- Question: What ends the 'end'

```
!$omp parallel    ! Loosely structured block – no 'block' follows
  !$omp parallel  ! Strictly structured block – 'block' next
```

- `    block         ! → '!$omp end parallel' is optional`
- `      x = x + 1`
- `    end block`
- `!$omp end parallel ! ← ends outer or inner 'parallel'?`

- OpenMP 5.1: ?
- OpenMP 5.2 added: 'An *end-directive* that immediately follows a directive ... is always paired with that directive.'
  → Applies to inner
  → An additional '!$omp end parallel' is missing

OMP_OFFLOAD_TARGET=mandatory (1/4)
When to Actually to Abort with an Error

OpenMP has (5.0 + 5.1)

"The **mandatory** value specifies that program execution is terminated if a device construct or device memory routine is encountered and the device is not available or is not supported by the implementation."

Real-world situation

- System with installed GPU, CUDA installed but no device available due to kernel issue (`cuInit: no CUDA-capable device is detected`)

→ omp_get_num_devices() == 0 (→ only host)

Question: Should fail with "mandatory" or not?

- GCC → host fallback/no fail – as no device exists, default device is the host
- LLVM → fails with the CUDA error
- User expectation: Hardware exists but does not work → should fail
  And if no hardware exists → no fail or still a fail?

SIEMENS

OMP_OFFLOAD_TARGET=mandatory (1/4)
When to Actually to Abort with an Error

OpenMP has (5.0 + 5.1)

"The **mandatory** value specifies that program execution is terminated if a device construct or device memory routine is encountered and the device is not available or is not supported by the implementation."

Real-world situation

- System with installed GPU, CUDA installed but no device available due to kernel issue (`cuInit: no CUDA-capable device is detected`)

→ omp_get_num_devices() == 0 (→ only host)

Question: Should fail with "mandatory" or not?

- GCC → host fallback/no fail – as no device exists, default device is the host
- LLVM → fails with the CUDA error
- User expectation: Hardware exists but does not work → should fail
  And if no hardware exists → no fail or still a fail?

Workaround

Environment variable: ?
OMP_DEFAULT_DEVICE=kind(gpu) would be useful, but does not exist in OpenMP ≤ 5.2

Code:

```
if (omp_get_default_device()
    == omp_get_initial_device())
  → error
```


SIEMENS

OMP_OFFLOAD_TARGET=mandatory (2/4)
When to Actually to Abort with an Error

Question: What if no non-host device is available and …

```
void foo () {}
#pragma omp declare target to (foo)

int main () {
  #pragma omp target if(false)
  foo ();       // Is this ok?
  omp_set_default_device (omp_get_initial_device ());
  #pragma omp target
  foo ();       // What about this?
  #pragma omp target device(omp_get_initial_device ())
  foo ();       // Or this?
  #pragma omp target device(omp_get_num_devices () + 42)
  foo ();       // This one is clearly an error
  if (omp_get_num_devices () == 3)
    {
      #pragma omp target device (1)
      foo ();   // This would be an error if we can't offload to device 1
    }
}
```


SIEMENS

OMP_OFFLOAD_TARGET=mandatory (3/4)
When to Actually to Abort with an Error

Solution in OpenMP 5.2

- **Definitions** — “the constant **omp_initial_device** can be used as an alias for the host device and the constant **omp_invalid_device** can be used to specify an invalid device number. A *conforming device number* is either a non-negative integer that is less than or equal to **omp_get_num_devices()** or equal to **omp_initial_device** or **omp_invalid_device**.”

- ***default-device-var* initialization** — “If *target-offload-var* is **mandatory** and the number of non-host devices is zero then the *default-device-var* is initialized to **omp_invalid_device**. Otherwise, the initial value is an implementation-defined non-negative integer that is less than or, if *target-offload-var* is not mandatory, equal to **omp_get_initial_device()**.”

- **OMP_OFFLOAD_TARGET=mandatory** — “The **mandatory** value specifies that the effect of any device construct or device memory routine that uses a device that is unavailable or not supported by the implementation, or uses a non-conforming device number, is as if the **omp_invalid_device** device number was used.”

OMP_OFFLOAD_TARGET=mandatory (4/4)
When to Actually to Abort with an Error

Solution in OpenMP 5.2 (con'd)

- **device_num** clause – “If the device-description evaluates to **omp_invalid_device**, runtime error termination is performed.”

- **Device Memory Routines:** “If the *device_num*, *src_device_num*, or *dst_device_num* argument of a device memory routine has the value **omp_invalid_device**, runtime error termination is performed.”

- **Definition:** “When *runtime error termination* is performed, the effect is as if an **error** directive for which *sev-level* is **fatal** and *action-time* is **execution** is encountered.”

Spec Work from Our Side

Mainly trying to fix issues and improve wording

- Usually found when implementing a feature and looking at the spec
- Sometimes found by chance or forwarding issues reported to us
- Taking care of issues found during the (sub)committee discussion

Recent examples (on going)

- Issues in Fortran part related to 5.1-added conditions support in ‘atomic’
- has_device_addr – trying to clarify semantics
- Extend OMP_DEFAULT_DEVICE (cf. mandatory discussion)

If you find a potential issue, bug, missing feature

- Check newer version of the spec – could be a bug which was fixed
- Contact some ARB member to take care of the bug (or participate if your org is already a member – or asked your org to become a member)

SIEMENS

Agenda

- GCC Overview
- OpenMP 5.0 + 5.1 Support
- Specification Corner Cases
- Command-line Options & Tricks
  - GCC Command Line Options for Offloading
  - Tricks & Tips – nvptx
  - Tricks & Tips – GCN
- Conclusion

GCC Command Line Options for Offloading

Solution in OpenMP 5.2 (con'd)

- Generates offload code by default for OpenMP target regions with -**fopenmp** (as configured; for instance for both nvptx and AMD GCN)

- **-foffload=<disable|default|*target-list*>** – restrict to those device types

- **-foffload-options=<*option*> or -foffload-options=<*target*>=<*options*>**
  specify option to the offload compiler
  Typical examples:
  -foffload-options=-lgfortran -foffload-options=-lm
  -foffload-options="-lgfortran -lm" -foffload-options=nvptx-none=-latomic
  -foffload-options=amdgcn-amdhsa=-march=gfx906 -foffload-options=-lm

- Verbose optimization pass diagnostic: **-fopt-info**[-*options*[=*filename*]]
  Example: -foffload-options=-fopt-info-loop-missed -fopt-info-omp-missed

Tricks & Tips
nvptx

JIT

GCC generates generic code, which is just-in-time compiled by CUDA
at startup – and cached in the user's directory.
→ https://developer.nvidia.com/blog/cuda-pro-tip-understand-fat-binaries-jit-caching/
→ CUDA_CACHE_{DISABLE,MAXSIZE,PATH}

GCC compile flags for nvptx

Usually not needed due to JIT

https://gcc.gnu.org/onlinedocs/gcc/Nvidia-PTX-Options.html

Possible exceptions:

- -mptx=N.N (PTX ISA version), -misa=sm_XX
- "illegal memory access was encountered" – generic error; could be stack issue, if so: -foffload=-msoft-stack-reserve-local=… might help.
  Default 128 byte (note: multiplied by sm_count×thread_max ~ 20000)

Hardware Specific Compilation

Native code for the specified GCN hardware is generated.

Use, e.g., -foffload-options=-march=fiji (GCN3, gfx803 – the default)
or for GCN5 GPUs: gfx900 (VEGA 10), gfx906 (VEGA 20), or gfx908.

ROCGDB

Offloading debugging is supported with ROCGDB.
→ Slides: https://linuxplumbersconf.org/event/11/contributions/997/
→ Video via ↑ or https://webinars.sw.siemens.com/debugging-offloaded-kernels-on-amd

Conclusion

OpenMP in GCC

- OpenMP 5.0 mostly supported + intial 5.1 support in GCC 12

Planned and/or useful

- GCC 13: Most of OpenMP 5.0, more of 5.1
- Improve device support: unified shared memory, performance, …
- Diagnostic, documentation improvements

Specification

- Large – and the devil is in the details

Communitiy effort

- Both the spec and the compiler depends on feedback, support, and work of users, developers (paid and hobby), and vendors

SIEMENS

Acknowledgement

This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725

Disclaimer