GCC/OpenMP Update

Tobias Burnus
Catherine Moore
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
GCC Release Cycle

<table>
<thead>
<tr>
<th>Date</th>
<th>Release</th>
<th>Development Branch</th>
</tr>
</thead>
<tbody>
<tr>
<td>April 2021</td>
<td>GCC 11</td>
<td>OG11 Branch</td>
</tr>
<tr>
<td>Spring 2022</td>
<td>GCC 12</td>
<td>OG12 Branch</td>
</tr>
<tr>
<td>Spring 2023</td>
<td>GCC 13</td>
<td>OG13 Branch</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>OpenMP Revision</th>
<th>Support Level</th>
<th>NVIDIA Offloading</th>
<th>AMD Offloading</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenMP 4.5</td>
<td>Fully Supported</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>OpenMP 5.0</td>
<td>Partial Support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>OpenMP 5.1</td>
<td>Minimal Support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

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
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:
• Development branch: https://gcc.gnu.org/onlinedocs/libgomp/

General help with GCC
• mailing list: https://gcc.gnu.org/pipermail/gcc-help/
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, Planned 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)
## OpenMP 5.0 Features Supported in GCC 12

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterators</td>
<td><code>omp_fulfill_event</code> runtime routine</td>
</tr>
<tr>
<td><code>target-offload-var ICV</code> and <code>OMP_TARGET_OFFLOAD</code> env variable</td>
<td><code>reduction</code> and <code>in_reduction</code> clauses on <code>taskloop</code> and <code>taskloop simd</code> constructs</td>
</tr>
<tr>
<td>Nested-parallel changes to <code>max-active-levels-var ICV</code></td>
<td><code>taskloop</code> construct cancelable by <code>cancel</code> construct</td>
</tr>
<tr>
<td><code>teams</code> construct outside an enclosing target region</td>
<td><code>mutexinouset</code> <code>dependence-type</code> for <code>depend</code> clause</td>
</tr>
<tr>
<td><code>!=</code> as relational-op in canonical loop form for C/C++</td>
<td>Predefined memory spaces, memory allocators, allocator traits</td>
</tr>
<tr>
<td><code>nonmonotonic</code> as default loop schedule modifier for worksharing-loop constructs</td>
<td>Memory management routines</td>
</tr>
<tr>
<td><code>Clauses if, nontemporal and order(concurrent) in simd</code> construct</td>
<td><code>use_device_addr</code> clause on target data</td>
</tr>
<tr>
<td><code>atomic</code> constructs in <code>simd</code></td>
<td>Implicit <code>declare target</code> directive</td>
</tr>
<tr>
<td><code>loop</code> construct</td>
<td>C/C++‘s lvalue expressions in <code>depend</code> clauses</td>
</tr>
<tr>
<td><code>order(concurrent)</code> clause</td>
<td>Nested <code>declare target</code> directive</td>
</tr>
<tr>
<td><code>scan</code> directive and <code>in_scan</code> modifier for the <code>reduction</code> clause</td>
<td>Combined <code>master</code> constructs</td>
</tr>
<tr>
<td><code>in_reduction</code> clause on <code>task</code> constructs</td>
<td><code>depend</code> clause on <code>taskwait</code></td>
</tr>
<tr>
<td><code>task_reduction</code> clause with <code>taskgroup</code></td>
<td>Weak memory ordering clauses on <code>atomic</code> and <code>flush</code> construct</td>
</tr>
<tr>
<td><code>task</code> modifier to <code>reduction</code> clause</td>
<td><code>depojb</code> construct and <code>depend</code> objects</td>
</tr>
<tr>
<td><code>affinity</code> clause to <code>task</code> construct</td>
<td>Lock hints were renamed to synchronization hints</td>
</tr>
<tr>
<td><code>detach</code> clause to <code>task</code> construct</td>
<td><code>conditional</code> modifier to <code>lastprivate</code> clause</td>
</tr>
<tr>
<td><code>close</code> <code>map-type-modifier</code></td>
<td><code>defaultmap</code> extensions</td>
</tr>
<tr>
<td><code>omp_get_supported_active_levels</code> routine</td>
<td>Runtime routines and environment variables to display runtime thread affinity information</td>
</tr>
<tr>
<td><code>omp_pause_resource</code> and <code>omp_pause_resource_all</code> runtime routines</td>
<td><code>omp_get_device_num</code> runtime routine</td>
</tr>
<tr>
<td>Supporting C++‘s range-based for loop</td>
<td></td>
</tr>
</tbody>
</table>
### OpenMP 5.1 Features Supported in GCC 12

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenMP directive as C++ attribute specifiers</td>
<td></td>
</tr>
<tr>
<td><strong>nothing</strong> directive</td>
<td></td>
</tr>
<tr>
<td><strong>error</strong> directive</td>
<td></td>
</tr>
<tr>
<td><strong>masked</strong> construct</td>
<td></td>
</tr>
<tr>
<td><strong>scope</strong> directive</td>
<td></td>
</tr>
<tr>
<td><strong>strict</strong> modifier in the <strong>grainsize</strong> and <strong>num_tasks</strong> clauses of the <strong>taskloop</strong> construct</td>
<td></td>
</tr>
<tr>
<td><strong>thread_limit</strong> clause to <strong>target</strong> construct</td>
<td></td>
</tr>
<tr>
<td>Extensions to the <strong>atomic</strong> directive</td>
<td></td>
</tr>
<tr>
<td><strong>seq_cst</strong> clause on a <strong>flush</strong> construct</td>
<td></td>
</tr>
<tr>
<td><strong>private</strong> and <strong>firstprivate</strong> argument to <strong>default</strong> clause in C and C++</td>
<td></td>
</tr>
<tr>
<td><strong>omp_set_num_teams</strong>, <strong>omp_set_teams_thread_limit</strong> runtime routines</td>
<td></td>
</tr>
<tr>
<td><strong>omp_get_max_teams</strong>, <strong>omp_get_teams_thread_limit</strong> runtime routines</td>
<td></td>
</tr>
<tr>
<td><strong>omp_calloc</strong>, <strong>omp_realloc</strong> runtime routines</td>
<td></td>
</tr>
<tr>
<td><strong>omp_aligned_alloc</strong> and <strong>omp_aligned_calloc</strong> runtime routines</td>
<td></td>
</tr>
<tr>
<td><strong>omp_allocattr_key_t</strong> enum: <strong>omp_atv_serialized</strong> added, <strong>omp_atv_default</strong> changed</td>
<td></td>
</tr>
<tr>
<td><strong>OMP_PLACES</strong> syntax extensions</td>
<td></td>
</tr>
<tr>
<td><strong>OMP_NUM_TEAMS</strong> and <strong>OMP_TEAMS_THREAD_LIMIT</strong> environment variables</td>
<td></td>
</tr>
<tr>
<td>Support of strictly structured blocks in Fortran</td>
<td></td>
</tr>
<tr>
<td>Support of structured block sequences in C/C++</td>
<td></td>
</tr>
<tr>
<td><strong>unconstrained</strong> and <strong>reproducible</strong> modifiers on order clause</td>
<td></td>
</tr>
<tr>
<td><strong>omp_display_env</strong> runtime routine</td>
<td></td>
</tr>
<tr>
<td>Feature</td>
<td>Development Branch Support</td>
</tr>
<tr>
<td>--------------------------------------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>Array shaping</td>
<td>No</td>
</tr>
<tr>
<td>Array sections with non-unit strides in C and C++</td>
<td>No</td>
</tr>
<tr>
<td>metadirective directive</td>
<td>Yes</td>
</tr>
<tr>
<td>Collapse of associated loops that are imperfectly nested loops</td>
<td>No</td>
</tr>
<tr>
<td>allocate directive</td>
<td>Yes</td>
</tr>
<tr>
<td>Discontiguous array section with target update construct</td>
<td>No</td>
</tr>
<tr>
<td>C/C++’s lvalue expressions in to, from, and map clauses</td>
<td>No</td>
</tr>
<tr>
<td>declare mapper directive</td>
<td>No</td>
</tr>
<tr>
<td>OMPT interface</td>
<td>No</td>
</tr>
<tr>
<td>OMPD interface</td>
<td>No</td>
</tr>
</tbody>
</table>
## OpenMP 5.1 Features Planned for GCC 13 – and beyond

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>omp_all_memory</code></td>
<td>reserved locator</td>
</tr>
<tr>
<td><code>target_device trait</code> in OpenMP Context</td>
<td><code>omp_target_memcpy_async</code> runtime routine</td>
</tr>
<tr>
<td><code>target_device</code> selector set in context selectors</td>
<td><code>omp_target_memcpy_rect_async</code> runtime routine</td>
</tr>
<tr>
<td>C/C++'s <code>declare variant</code> directive: elision support of preprocessed code</td>
<td><code>omp_get_mapped_ptr</code> runtime routine</td>
</tr>
<tr>
<td><code>declare variant</code>: new clauses <code>adjust_args</code> and <code>append_args</code></td>
<td><code>omp_get_mapped_ptr</code> runtime routine</td>
</tr>
<tr>
<td><code>dispatch</code> construct</td>
<td><code>omp_get_mapped_ptr</code> runtime routine</td>
</tr>
<tr>
<td>device-specific ICV settings the environment variables</td>
<td><code>omp_state_t enum: omp_state_wait_barrier_implementation</code></td>
</tr>
<tr>
<td><code>assume</code> directive</td>
<td><code>omp_state_t enum: omp_state_wait_barrier_teams</code></td>
</tr>
<tr>
<td>Loop transformation constructs</td>
<td><code>ompt_callback_target_data_op_emi_t</code>, <code>ompt_callback_target_emi_t</code></td>
</tr>
<tr>
<td><code>has_device_addr</code> clause to target construct</td>
<td><code>ompt_callback_target_map_emi_t</code> and <code>ompt_callback_target_submit_emi_t</code></td>
</tr>
<tr>
<td>iterators in <code>target update</code> motion clauses and <code>map</code> clauses</td>
<td><code>ompt_callback_error_t</code> type</td>
</tr>
<tr>
<td>indirect calls to the device version of a procedure or function in target regions</td>
<td><code>nowait</code> clause in <code>taskwait</code> directive</td>
</tr>
<tr>
<td><code>inoutset</code> argument to the <code>depend</code> clause</td>
<td><code>interop</code> directive</td>
</tr>
<tr>
<td><code>present</code> argument to <code>defaultmap</code> clause</td>
<td><code>omp_interop_t</code> object support in runtime routines</td>
</tr>
<tr>
<td><code>omp_target_is_accessible</code></td>
<td>runtime routine</td>
</tr>
</tbody>
</table>
### OpenMP 5.0 Features Partially Supported in GCC 12

<table>
<thead>
<tr>
<th>Directive</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>declare variant</code></td>
<td>SIMD traits not handled correctly</td>
</tr>
<tr>
<td><code>requires</code> directive</td>
<td>Only fulfillable requirements are <code>atomic_default_mem_order</code> and <code>dynamic_allocators</code></td>
</tr>
<tr>
<td>Non-rectangular loop nests</td>
<td>Fortran support missing, planned for GCC 13</td>
</tr>
<tr>
<td><code>in_reduction</code> clause on target constructs</td>
<td><code>nowait</code> is only a stub</td>
</tr>
<tr>
<td><code>ancestor</code> modifier on device clause</td>
<td>Reverse offload unsupported</td>
</tr>
</tbody>
</table>

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 unsupported, planned for GCC 13

### OpenMP 5.1 Features Partially Supported in GCC 12

<table>
<thead>
<tr>
<th>Directive</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>align</code> clause/modifier in <code>allocate</code> directive/clause and <code>allocator</code> directive</td>
<td>Fortran support missing, planned for GCC 13</td>
</tr>
</tbody>
</table>

Planned for GCC 13:
- Unified shared memory support with NVIDIA GPUs
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?
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)
  \( \rightarrow \) omp_get_num_devices() == 0 (\( \rightarrow \) only host)

Question: Should fail with “mandatory” or not?

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

Workaround

Environment variable: ?

\( \text{OMP_DEFAULT_DEVICE=kind(gpu)} \)

would be useful, but does not exist in OpenMP \( \leq 5.2 \)

Code:

if (omp_get_default_device() == omp_get_initial_device())
  \( \rightarrow \) 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
  }
}
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.”
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)
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.


→ 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)
**Tricks & Tips**

**GCN**

**Hardware Specific Compilation**

Native code for the specified GCN hardware is generated.

Use, e.g., `-offload-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/](https://linuxplumbersconf.org/event/11/contributions/997/)
→ Video via ↑ or [https://webinars.sw.siemens.com/debugging-offloaded-kernels-on-amd](https://webinars.sw.siemens.com/debugging-offloaded-kernels-on-amd)
Conclusion

OpenMP in GCC
• OpenMP 5.0 mostly supported + initial 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

Community effort
• Both the spec and the compiler depends on feedback, support, and work of users, developers (paid and hobby), and vendors
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
© Siemens 2022

Subject to changes and errors. The information given in this document only contains general descriptions and/or performance features which may not always specifically reflect those described, or which may undergo modification in the course of further development of the products. The requested performance features are binding only when they are expressly agreed upon in the concluded contract.

All product designations may be trademarks or other rights of Siemens AG, its affiliated companies or other companies whose use by third parties for their own purposes could violate the rights of the respective owner.