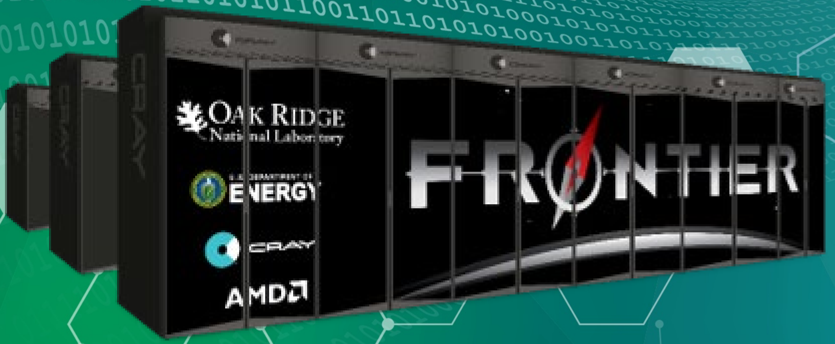
OpenMP Offload in GCC: Status and Plans

David E. Bernholdt and Wael R. Elwasif

Programming Environment and Tools
Oak Ridge Leadership Computing Facility
and Computer Science and Mathematics Division
Oak Ridge National Laboratory

Tobias Burnus
Siemens Digital Industries Software
(formerly Mentor Graphics)



Who?

- A partnership between the Oak Ridge Leadership Computing Facility and Siemens Digital Industries Software to build out GCC’s implementation of OpenMP offloading
  - Also OpenACC and selected Fortran 2018 capabilities

DOE Office of Science Advanced Scientific Computing Research Facilities

Providing the Facilities – High-End and Leadership Computing

Leadership Computing Centers at Oak Ridge National Laboratory (OLCF) and Argonne National Laboratory (ALCF)

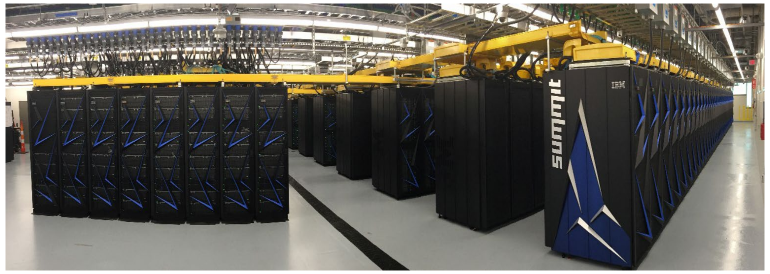
- Deliver the highest computational capability
- Open to national and international researchers, including industry
- Not constrained by existing DOE or Office of Science funding or topic areas
- Approximately 1,000 users and 50-60 projects at each center, each year

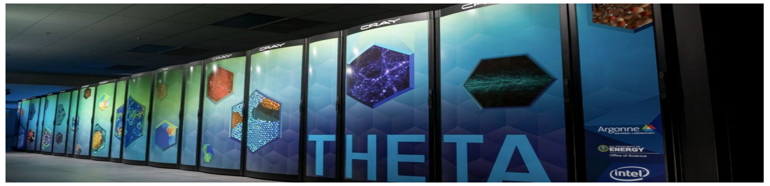
National Energy Research Scientific Computing Center (NERSC) Lawrence Berkeley National Laboratory

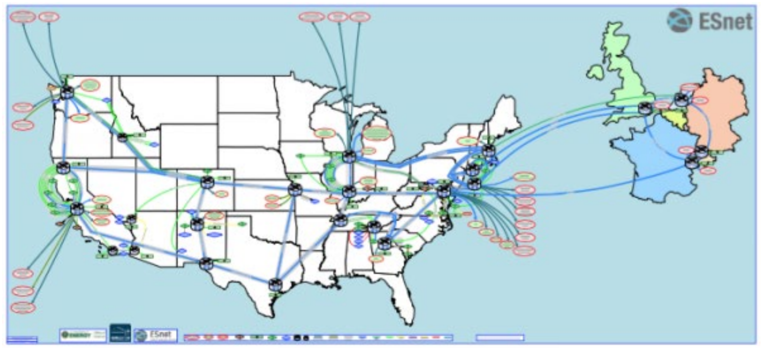
- Delivers high-end capacity computing to entire DOE SC research community
- Over 6,000 users and 800 projects

Energy Sciences Network (ESnet)

- Linking it all together

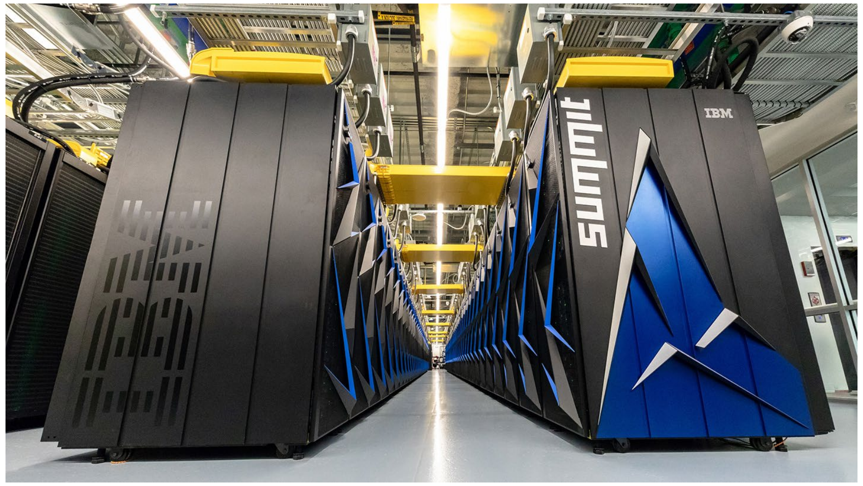






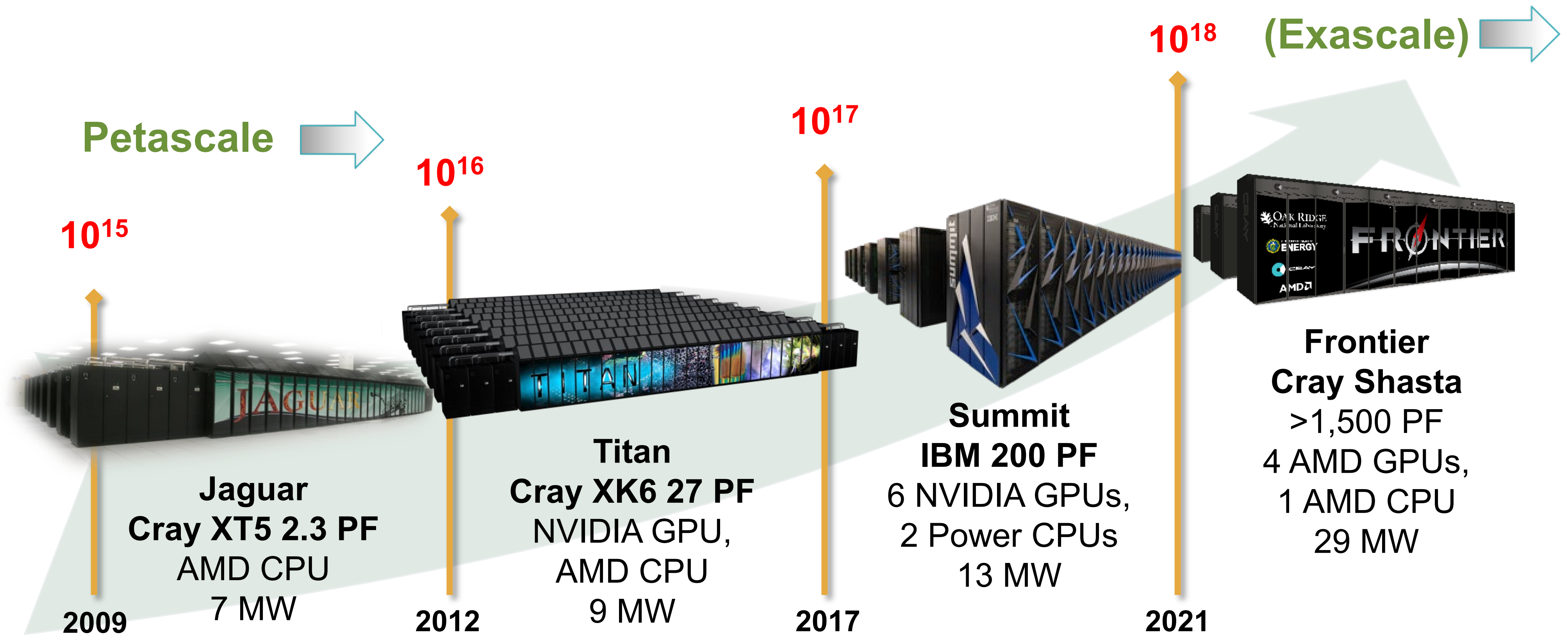
DOE Leadership Computing Facility

- **The Leadership Computing Facility** is a collaborative, multi-lab, DOE/SC initiative ranked top domestic priority; 2 centers/2 architectures to address diverse and growing computational needs of the scientific community
- **Mission:** Provide an ecosystem, including partnering opportunities, that enables unsurpassed capability computing opportunities and the associated science and engineering breakthroughs
- Administer and support two highly competitive user allocation programs (INCITE, ALCC)
  - Innovative and Novel Computational Impact on Theory and Experiment (INCITE)
  - ASCR Leadership Computing Challenge (ALCC)
  - Computational allocations typically 100 times greater than routinely available for university, laboratory, and industrial scientific and engineering environments



Oak Ridge Leadership Computing Facility Roadmap to Exascale

Mission: Providing world-class computational resources and specialized services for the most computationally intensive global challenges for researchers around the world.



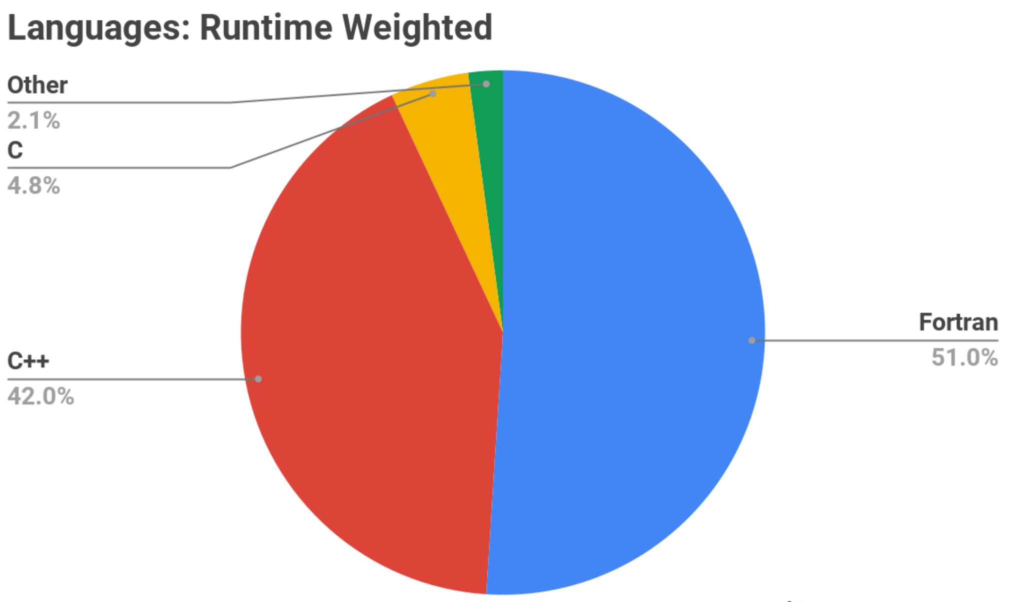
Jaguar
Cray XT5 2.3 PF
AMD CPU
7 MW

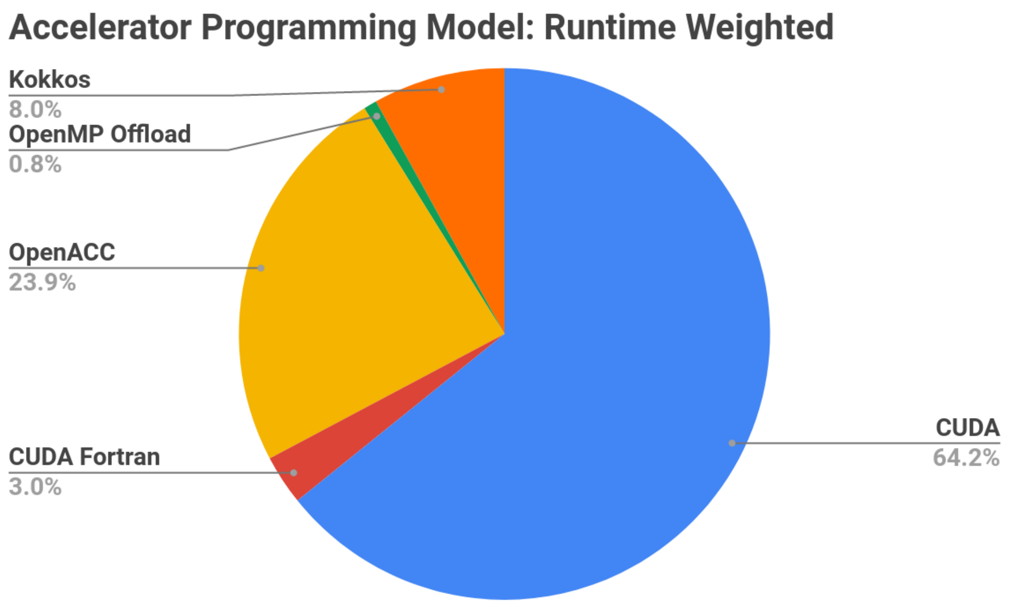
Titan
Cray XK6 27 PF
NVIDIA GPU,
AMD CPU
9 MW

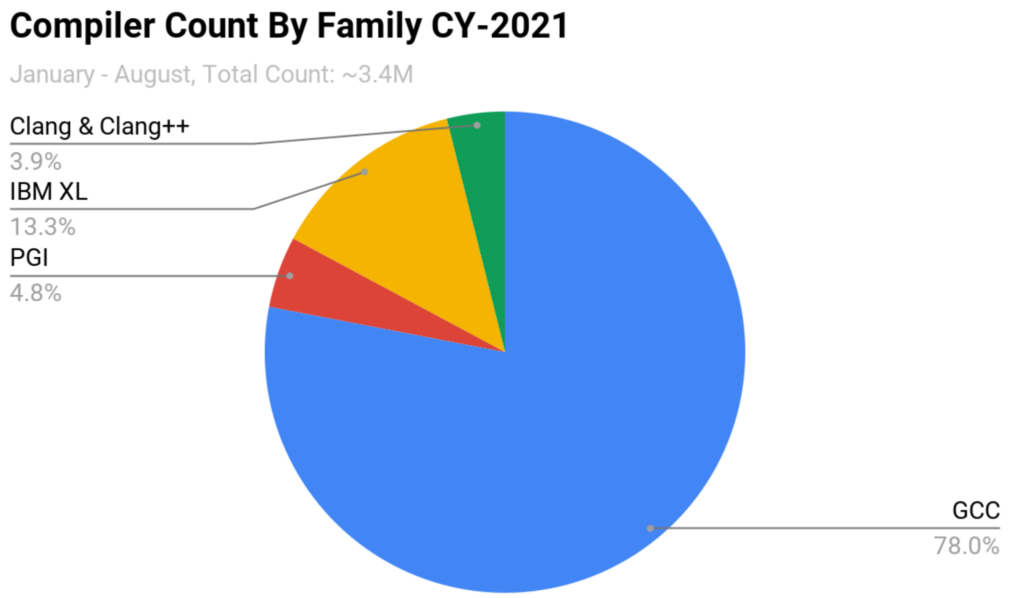
Summit
IBM 200 PF
6 NVIDIA GPUs,
2 Power CPUs
13 MW

Frontier
Cray Shasta
>1,500 PF
4 AMD GPUs,
1 AMD CPU
29 MW

Programming Summit







Statistics for CY2021 to date

Data and graphs courtesy of Reuben Budiardja and Bill Renaud

Frontier Programming Environment

Items in green are also available on Summit

- Compilers Offered
  - Cray PE (C/C++ LLVM-based; Cray Fortran)
  - AMD ROCm (LLVM-based)
  - LLVM (provided by OLCF)
  - GCC (provided by OLCF (latest) and HPE/Cray (older))
- Programming Languages & Models Supported (in which compilers)
  - C, C++, Fortran (all)
  - OpenACC (GCC) 2.6 substantially complete, 2.7 planned
  - OpenMP (all)
  - HIP (Cray, AMD) – New: Cray has added HIP support to CPE
  - Kokkos/RAJA (all)
- Transition Paths
  - CUDA: semi-automatic translation to HIP
  - CUDA Fortran: HIP kernels called from Fortran (a more portable approach)
    - CUDA Fortran kernels need to be translated to C++/HIP (manual process)
    - Fortran bindings to HIP and ROCm libraries and HIP runtime available through AMD's hipfort project

The OLCF-Siemens Collaboration – Background

- GCC is popular at OLCF, and elsewhere
- ORNL started working with NVIDIA and (then) Mentor Graphics on OpenACC in GCC in 2016
- OpenACC was the first community-driven standard directive language available for programming of GPU accelerators
  - OLCF helped launch in the Titan time frame
  - Important for Titan and Summit users
  - Desire to have multiple production-quality implementations available
- As OpenMP offload specification matured, we also wanted multiple implementations of it
  - Need for AMD GPU support for Frontier
  - Expanding for NVIDIA also supports Summit

Overview of Approach and Plans

- Tasks are used to deliver functionality
- Directed Services are used to address performance and correctness issues

Tasks Contracted or Planned

- OpenACC kernels improvements, 2.7 features
- Select Fortran 2018 interoperability features
- OpenMP 5.0 (most features)
- OpenMP 5.1 (most features)
- Supporting NVIDIA (V100) and AMD (MI100) accelerators

Not Included

- OMPT
  - Deferred in favor of core programming features
- OMPD
  - Not used by Arm DDT, ORNL's primary scalable debugger
- Base language support not already present in GCC

GCC
The GNU Compiler Collection

- Compiler for C, C++, Fortran, Ada, D, go, …
  - C17 (steps to C2x), C++20 (steps to C++23)
  - Fortran 2008+coarray/interop TS (mostly), initial F2018
- Supported archs: aarch64, alpha, arc, arm, avr, bfin, …

- Annual major releases around late spring (April, May)
- Unregular .dot releases for the release branches
  (However, Linux distros use the git branch directly)

- OpenSource software, developed by paid and unpaid developers
- Bug reports & testcases welcome
  … with the usual timing and resource issues of community support
- Contributions – patches welcome
  Larger ones require either FSF copyright assignment
  or, new, just a "signed-of-by", certifying the right to submit

https://gcc.gnu.org

OpenMP & Offloading
Support in GCC

GCC 9 – 2019

OpenMP 4.5 (C/C++, Fortran mostly), some OpenMP 5.0 support.
OpenACC 2.5 mostly. Offloading to nvptx (Nvidia).

GCC 10 – 2020

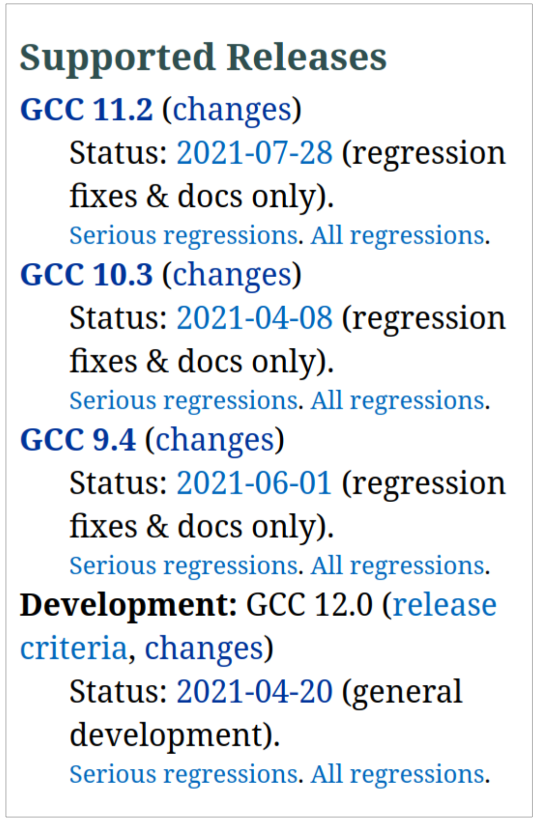
Extended OpenMP 5.0 support (e.g. scan + loop directive in C/C++, initial support for declare variant (C/C++), use_device_addr, …).
OpenACC 2.6. Offloading to AMD Radeon (GCN: fiji, gfx900/gfx906)

GCC 11 – 2021

Extended OpenMP 5.0 support, especially for Fortran. GCN: gfx908.

GCC 12 – mainline – 2022

First OpenMP 5.1 features: directives as C++ attributes, masked and scope construct, nothing and error directive. Extended OpenMP 5.0 (affinity clause, loop directive in Fortran, …) – on going.
GCN: Uses more teams & threads.



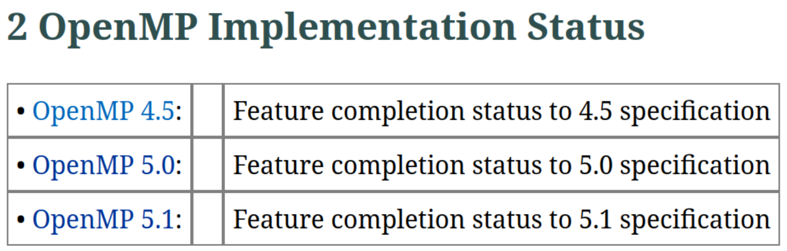
Supported Releases

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: 2021-04-20 (general development).
Serious regressions. All regressions.



2 OpenMP Implementation Status

• OpenMP 4.5:	Feature completion status to 4.5 specification
• OpenMP 5.0:	Feature completion status to 5.0 specification
• OpenMP 5.1:	Feature completion status to 5.1 specification

https://gcc.gnu.org/onlinedocs/libgomp/

SIEMENS

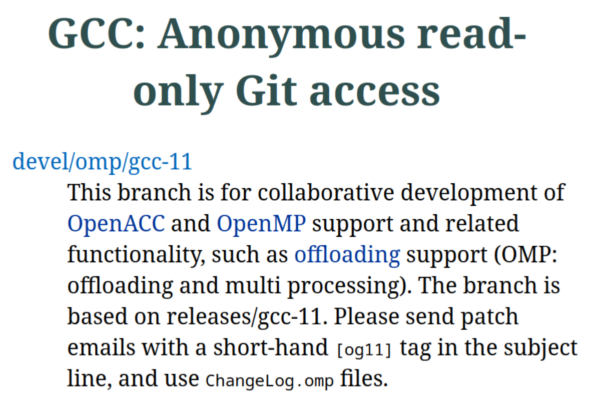
GCC Branches
Public Git Branches

Release Branches

- release/gcc-11 (latest), release/gcc-10, …

The OG11 Branch – devel/omp/gcc-11

- Siemens's OpenMP/offloading GCC branch
- Based on the GCC 11 release branch
- Contains OpenMP/OpenACC/offloading commits from GCC 12 / mainline and some additional improvement not yet on mainline



GCC: Anonymous read-only Git access

devel/omp/gcc-11
This branch is for collaborative development of OpenACC and OpenMP support and related functionality, such as offloading support (OMP: offloading and multi processing). The branch is based on releases/gcc-11. Please send patch emails with a short-hand `[og11]` tag in the subject line, and use `ChangeLog.omp` files.

https://gcc.gnu.org/git.html

SIEMENS

Offloading GCC Compiler
Obtaining an Offloading Compiler

Linux Distributions

Distros provide optional packages to support offloading – simply install:

- Debian/Ubuntu, gcc-11-offload-{nvptx,amdgcn}
- (open)SUSE: cross-{nvptx,amdgcn}-gcc11
- Red Hat/Fedora: {gcc,libgomp}-offload-nvptx (currently no amdgcn)

Ask your HPC center or IT to install one

Build GCC yourself

- Simple without offloading: https://gcc.gnu.org/install/
- More complex with offloading: https://gcc.gnu.org/wiki/Offloading

Fat files: a compiled progam can support, e.g., both nvptx and amdgcn.
Building for nvptx/amdgcn as embedded target is also possible.

SIEMENS

Offload Code Generation
nvptx

nvptx (Nvidia)

- GCC generates hardware-independent nvptx code, only
- Used feature set depends on selected
  PTX ISA (≈ CUDA) version (-mptx=N.N) and target (-misa=sm_XX).
- Embeds nvptx directly
- When run, CUDA JIT compiles nvptx for actual hardware
  → result is cached (see CUDA documentation)
- No CUDA required to create the binary
  but ptxas used for verification when available

Offload Code Generation
AMD Radeon (GCN)

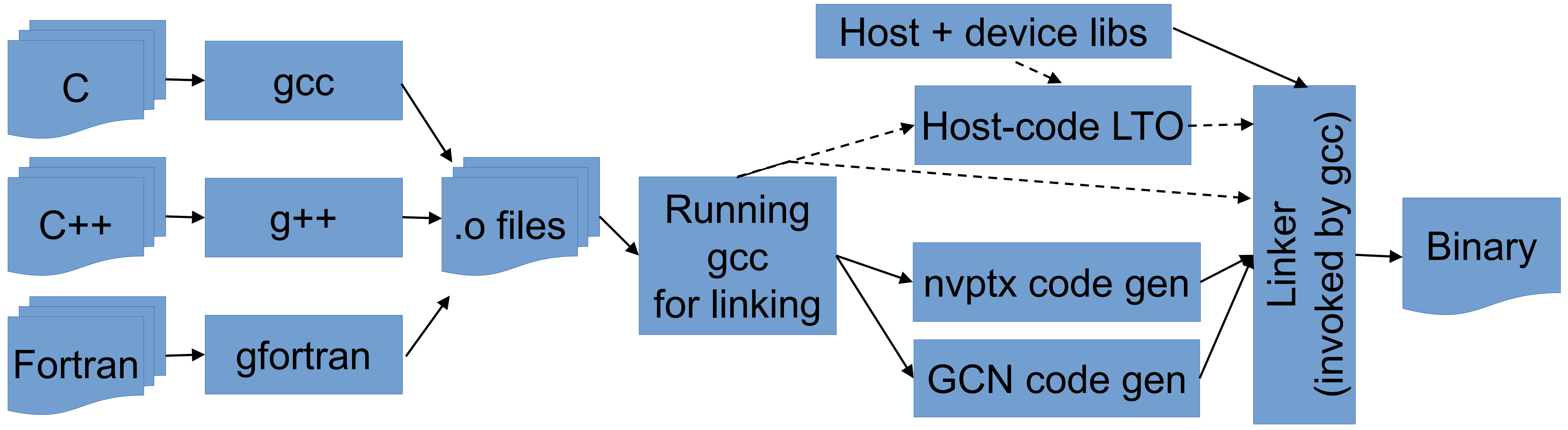
AMD Radeon (GCN)

- GCC generates assembler code for the specified hardware
  (fiji, gfx900, gfx906, gfx908)
- Only runs on that hardware
  no fat-file support to support multiple GCN targets in one file
- Calls llvm-mc assembler + lld (no GCN support in Binutils/GNU “as”/“ld”.)

→ **Talk:** Andrew Stubbs (Siemens), “**Debugging** offloaded kernels on AMD GPUs” using GDB, Thu, 23 Sept, 7:30am PDT/18:30 CEST. GNU Tools @ Linux Plumber Conference, https://LinuxPlumbersConf.org/

SIEMENS

Compilation for Offloading
Internal Handling of Compilation and Linking



The object file contains the normal host compilation + split-off target-regions and offload functions in a generic format (LTO)

All code generation steps for a specific offload target only done at link time. Offload code: by-target assembly/linking, result is then linked together with host code.
[Code gen: Also creates a table with func and global vars ptrs for matching host and device funcs/vars.]

- For which offload target code is generated: decision deferred to link time
- Input files are only parsed once
- Does not permit to use different header files for different offload compilers

SIEMENS

Execution with Offloading
Initialization and Device Handling

Startup

- libgomp loads plugins for the supported devices (nvptx, gcn, …)
  – each returning the number of available devices.

On-device execution

- Once a device is required (e.g. “target enter data”):
  → This Device is initialized
  → (nvptx only) JIT compilation the remote functions

Levels of Parallelism
Offloading

- All three levels used: teams, threads and simd
- SIMD / vectorized loops map to thread/work item
- teams + parallel map to warps/wavefronts,
  OpenMP teams uses threadpool of size #teams
- **nvptx**
  warp_size ≡ 32
  CUDA kernel launched: dim={#teams,1,1}, blocks={#threads,warp_size,1}.
  *Use GOMP_DEBUG=1 and grep for "kernel.*launch"*
- **GCN***
  #teams = (#CU|num_teams), #threads = (16**|num_threads ≤ 16**)
  #workitems = 64
  Hardware: max 40 workgroups/CU, 16 wavefronts/workgroup
  Current impl.: 80 scalars registers and 24 vector registers in non-kernel functions (proc calling ABI) / Kernel itself: as many as register pressure demands (#teams + #threads scaled down if registers exhausted)

OpenMP	OpenACC
teams	gang
threads (parallel)	worker
simd	vector

Nvidia	AMD
streaming multiprocessor	compute unit (CU)
thread block	work group
warp	wavefront
thread	work item (thread)

* as in OG11, in GCC 12 (similar, soon: same). GCC ≤ 11: one thread/team only / ** for gfx900: 4

Summary

- ORNL has partnered with Siemens Digital Industries Software to build out the OpenMP offload capabilities in GCC
- With specific support for NVIDIA V100 and AMD MI100 GPU accelerators
  - V100 used in Summit
  - MI100 precursor to the GPU used in Frontier
- Work is in progress, and will be for some time
- Results are on a publicly accessible branch and upstreamed to mainline GCC
  - See also the “Tips & Tricks” at the end of this presentation

Opportunities to Contribute

- OpenMP offload is a big lift
- GCC is a community product
- You can leverage the work ORNL and Siemens are doing to further improve GCC…
  - Support for additional platforms
  - Supporting features not prioritized by ORNL
  - Identifying, reporting, and addressing correctness and performance issues
- We are happy to coordinate with you

Tricks & Tips
Reported OpenMP Version and Compiler Flags

_OPENMP / openmp_version

Currently, defined to 201511 (OpenMP 4.5) – as with most compilers
Hence, it cannot be used to detect a mostly 5.0 supporting compiler.

Compiler flags for offloading

-foffload=disable, -foffload=amdgcn-amdhsa, …
Whether to compile for an offload target – and for which one.

-foffload-options=-lm -foffload-options=nvptx-none=-latomic
Pass arguments to the offload compiler, i.e. link math library
for all offload archs, but libatomic only for nvptx.
For Fortran: Some programs might require -foffload=-lgfortran
Before GCC 12: Use undocumented
-foffload=-lm -foffload=nvptx-none=-latomic instead.

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}

Nvptx compiler flags

Usually not needed due to JIT
https://gcc.gnu.org/onlinedocs/gcc/Nvidia-PTX-Options.html
Possible exceptions:

- "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)
- -mptx=N.N, -misa=sm_XX

Tricks & Tips
GCN

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.

Teams/Threads

Current implementation uses 16 wavefronts (= OpenMP's threads, OpenACC's worker) with 102 scalars, and 64 vectors.

New in GCC 12 / OG11: >1 thread per team, tuning of # of used wavefronts and workgroups

ROCGDB

Offloading debugging is supported with ROCGDB. Best use GCC 12.

Disclaimer

SIEMENS