GCC/OpenACC
GCC Branches (public)

- GCC 11: current stable release series
  - OpenACC 2.6 support
  - Code offloading to AMD (GCN) and Nvidia (nvptx) GPUs
- og11 development branch
  - GCC 11 release branch plus new features:
    - Backlog inherited from og10 branch, etc.
    - Current work
  - Provides early access to certain GCC 12+ features, on stable GCC 11 baseline
- GCC 12: next major release due spring 2022
  - In stabilization phase, as of yesterday
GCC/OpenACC
Last Year: Development Focus

- OpenACC 'kernels' work
  - Compiler "magic":
    - Array access delinearization
    - Scalar data privatization
    - Analyze 'loop' constructs with 'auto' clause, decide 'seq' vs. 'independent'
  - See talk at LPC¹, GNU Tools Track: *OpenACC "kernels" improvements* (Frederik Harwath)
    - [https://linuxplumbersconf.org/event/11/contributions/998/](https://linuxplumbersconf.org/event/11/contributions/998/)
    - [https://youtu.be/zUw0ZVXCwoM?t=12304s](https://youtu.be/zUw0ZVXCwoM?t=12304s)
  - Developed on private branch; now integrating into public og11 branch, later GCC mainline

- Revision and upstreaming of existing development branch work into GCC mainline

¹ Linux Plumbers Conference 2021, [https://linuxplumbersconf.org/](https://linuxplumbersconf.org/), virtual, week of 2021-09-20
GCC/OpenACC
Last Year: Revision/Upstreaming

Integrated into GCC mainline (for GCC 12):

- Decompose OpenACC 'kernels' constructs into parts, a sequence of compute constructs
- OpenACC worker parallelism for AMD GPUs
  - Execution state changes (neutering/broadcasting) as a GCC middle end transformation
  - Different approach from nvptx where it all happens in the back end
- Interactive debugging (GDB; ROCGDB) for GCN offloading
  - See demo at LPC¹, GNU Tools Track: *Debugging offloaded kernels on AMD GPUs* (Andrew Stubbs)
    - [https://linuxplumbersconf.org/event/11/contributions/997/](https://linuxplumbersconf.org/event/11/contributions/997/)
    - [https://youtu.be/q5itHU2T5xU?t=4396s](https://youtu.be/q5itHU2T5xU?t=4396s)
Integrated into GCC mainline (for GCC 12), continued:

- Bug fixing (such as OpenACC specification adherence), for example:
  - Data privatization/sharing at the OpenACC gang level: use GCN LDS, nvptx '.shared' memory
  - OpenACC/Fortran: strided array sections and components of derived-type arrays
  - OpenACC 'async' correctness
  - Keep up with occasional breaking changes of LLVM/HSA (GCN), PTX/CUDA (nvptx)
  - The usual miscellanea

- Code generation optimizations: middle end as well as GCN, nvptx back ends
- Diagnostics: '-Wopenacc-parallelism' to diagnose potentially suboptimal choices of OpenACC parallelism
GCC/OpenACC
Next Steps

- More revision/upstreaming of existing development branch work into GCC mainline
- Implement OpenACC 2.7 features (not yet scheduled)
  - Maybe YOU would like to hire us for OpenACC 3.0, 3.1, 3.2?
- Other items as prioritized by customers – existing and NEW
  - Bug fixing, performance enhancements, OpenACC features, interoperability, new offload devices, etc.
- ... and: continue working with OpenACC Technical Committee

Thanks!

Contact: <thomas_schwinge@mentor.com>
Disclaimer

© Siemens 2021

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.