International Workshop on Accelerator Programming Using Directives

WACCPD 2021: **Accelerator Programming Using Directives** pp 3–21

Can Fortran's 'do concurrent' Replace Directives for Accelerated Computing?

Miko M. Stulajter, Ronald M. Caplan & Jon A. Linker

Conference paper | First Online: 15 May 2022

Part of the Lecture Notes in Computer Science book series (LNPSE,volume 13194)

Abstract

Recently, there has been growing interest in using standard language constructs (e.g. C++'s Parallel Algorithms and Fortran's do concurrent) for accelerated computing as an alternative to directive-

based APIs (e.g. OpenMP and OpenACC). These constructs have the potential to be more portable, and some compilers already (or have plans to) support such standards. Here, we look at the current capabilities, portability, and performance of replacing directives with Fortran's `do concurrent` using a mini-app that currently implements OpenACC for GPU-acceleration and OpenMP for multi-core CPU parallelism. We replace as many directives as possible with `do concurrent`, testing various configurations and compiler options within three major compilers: GNU's `gfortran`, NVIDIA's `nvfortran`, and Intel's `ifort`. We find that with the right compiler versions and flags, many directives can be replaced without loss of performance or portability, and, in the case of `nvfortran`, they can all be replaced. We discuss limitations that may apply to more complicated codes and future language additions that may mitigate them. The software and Singularity/Apptainer containers are publicly provided to allow the results to be reproduced.

Keywords

accelerated computing **OpenMP** **OpenACC** **do concurrent**

standard language parallelism

Supported by NSF awards AGS 202815 and ICER 1854790, and NASA grant 80NSSC20K1582. This work used the Extreme Science and Engineering Discovery Environment (XSEDE) Bridges2 at the Pittsburgh Supercomputer Center through allocation TG-MCA03S014. It also used the DGX A100 system at the Computational Science Research Center at San Diego State University provided by NSF award OAC 2019194.

Notes

1. www.openmp.org.

2. www.openacc.org.

3. https://developer.nvidia.com/hpc-sdk.

4. https://gcc.gnu.org/.

5. https://flang.llvm.org.

6. https://developer.amd.com/amd-aocc.

7. https://www.ibm.com/products/xl-fortran-linux-compiler-power.

8. https://support.hpe.com/hpesc/public/docDisplay?docId=a00115296en_us&page=index.html.

9. Note that as this paper was going to press, HPE has indicated plans to support DC on GPUs.

10. https://releases.llvm.org/12.0.0/tools/flang/docs/DoConcurrent.html.

11. https://j3-fortran.org/doc/year/18/18-007r1.pdf.

12. https://j3-fortran.org/doc/year/21/21-007.pdf.

13. https://www.openacc.org/blog/announcing-openacc-31.

14. https://www.openmp.org/spec-html/5.1/openmp.html.

15. For `nvfortran 21.7`, it appears that setting the `-stdpar=gpu` flag implicitly sets the `-acc=gpu` option as well. This is an important consideration if one has OpenACC directives that should be ignored when using `-stdpar`.

16. www.predsci.com/papers/dc.

References

1. Balarac, G., et al.: AVBP and YALES2 portability, tuning and scalability on AMD EPYC 7002 Rome processors (2020)

2. Caplan, R.M., Downs, C., Linker, J.: Preparing photospheric magnetic field measurements for use in coronal and heliospheric models. In: AGU Fall Meeting Abstracts, vol. 2019, pp. SH43E–3389 December (2019)

3. Caplan, R.M., Mikić, Z., Linker, J.A., Lionello, R.: Advancing parabolic operators in thermodynamic MHD models: explicit super time-stepping versus implicit schemes with krylov solvers. J. Phys. Conf. Series **837**, 012016 (2017). https://doi.org/10.1088/1742-6596/837/1/012016

4. Caplan, R.M., Downs, C., Linker, J.A., Mikic, Z.: Variations in finite-difference potential fields. Astrophys. J. **915**(1), 44 (2021). https://doi.org/10.3847/1538-4357/abfd2f

5. Chandrasekaran, S., Juckeland, G.: OpenACC for Programmers: Concepts and Strategies. Addison-Wesley Professional (2017)

6. David Olsen, Graham Lopez, B.A.L.: Accelerating standard C++ with GPUs using stdpar (2021).https://developer.nvidia.com

/blog/accelerating-standard-c-with-gpus-using-stdpar/

7. Kurtzer, G.M., Sochat, V., Bauer, M.W.: Singularity: scientific containers for mobility of compute. PLOS ONE **12**(5), e0177459 (2017). https://doi.org/10.1371/journal.pone.0177459

8. Meyer, C.D., Balsara, D.S., Aslam, T.D.: A stabilized Runge-Kutta-Legendre method for explicit super-time-stepping of parabolic and mixed equations. J. Comput. Phys. **257**, 594–626 (2014). https://doi.org/10.1016/j.jcp.2013.08.021

9. Mikic, Z., Caplan, R.M., Linker, J.A., Stulajter, M.: Reproducibility package for running the DIFFUSE test cases from "Can Fortran's 'do concurrent' replace directives for accelerated computing" (2021). https://doi.org/10.5281/zenodo.5253520

10. Ozen, G., Lopez, G.: Accelerating Fortran do concurrent with GPUs and the NVIDIA HPC SDK (2020). https://developer.nvidia.com/blog/accelerating-standard-c-with-gpus-using-stdpar/

11. Van der Pas, R., Stotzer, E., Terboven, C.: Using OpenMP The Next

Step: Affinity. Tasking, and SIMD. MIT press, Accelerators (2017)

12. Towns, J., Cockerill, T., Dahan, M., Foster, I., Gaither, K., Grimshaw, A., Hazlewood, V., Lathrop, S., Lifka, D., Peterson, G.D., Roskies, R., Scott, J.R., Wilkins-Diehr, N.: XSEDE: accelerating scientific discovery. Comput. Sci. Eng. **16**(5), 62–74 (2014). https://doi.org/10.1109/mcse.2014.80

13. Xinman Tian, Kari Qi, M.L.: Practical examples of OpenMP offload to GPUs (2021).https://techdecoded.intel.io/essentials/3-quick-practical-examples-of-openmp-offload-to-gpus/

Author information

Authors and Affiliations

Predictive Science Inc., 9990 Mesa Rim Road Suite 170, San Diego, CA, 92121, USA

Miko M. Stulajter, Ronald M. Caplan & Jon A. Linker

Corresponding author

Correspondence to Miko M. Stulajter .

Editor information

Editors and Affiliations

Lawrence Berkeley National Laboratory, Berkeley, CA, USA

Sridutt Bhalachandra

Lawrence Berkeley National Laboratory, Berkeley, CA, USA

Christopher Daley

Oak Ridge National Laboratory, Oak Ridge, DE, USA

Verónica Melesse Vergara

Appendices

Appendix

Singularity/Apptainer containers: In order to test the latest compilers and to simplify setup of our library dependencies, we utilized Singularity/Apptainer containers. These containers allow one to run software in a containerized environment on any compatible system using only the container file.

Container setup: The containers were straight forward to setup and use for our timings. Two methods were used to create them. For `nvfortran` and `ifort`, a docker image of NVIDIA HPC SDK or Intel OneAPI HPC Toolkit was used to create a sandbox. For `gfortran`, a

similar sandbox with Ubuntu 21.04 was created and then `gfortran` was installed with the `apt-get` command. Once the sandboxes were created, the dependent libraries were installed. A sandbox is treated like a virtual machine, allowing us to edit and install new software into the container (note this requires `sudo` privileges). Once all the needed software is installed, the sandbox is converted to a `.sif` file which can be copied and run (without `sudo` privileges) on any other compatible machine with Singularity/Apptainer installed, but it can no longer be edited. However, the container is able to modify files outside itself, allowing us to compile and run the test cases. For GPU-accelerated runs, a special flag is needed when running the container depending on the vendor of GPU. For NVIDIA GPUs, the flag is `--nv`, while for AMD GPUs, the flag is `--rocm`. For more details on Singularity/Apptainer containers, see Ref. [7].

Container performance tests: Using containers can sometimes cause performance overhead. To ensure that using the Singularity/Apptainer containers does not cause significant overhead in our case, we ran two test cases on both a bare metal setup and with a container with the same compiler version (in this case `gfortran` 10.2). Table 12 shows timings of the test run using both the *Original* and *Serial* codes described in Sect. 3.2. We see that the runs using the container

perform nearly identical to those run on bare metal, allowing us to confidently use the containers for the runs in the paper.

Table 12. Timing results on a Bridges2 CPU compute node using `gfortran` 10.2 bare metal and form within a Singularity Container

Reproducibility package: The results in this paper can be reproduced using our reproducibility package hosted publicly at Ref. [9] and on our website[16]. The package contains three Singularity/Apptainer containers (for `gfortran`, `nvfortran`, and `ifort`), as well as all code versions, compiler options, and test cases. The package requires minimal customization (only specifying hardware-specific compiler options) of the main script, which can then be used to automatically run either all, or a subset, of runs from the paper. See the documentation in the package for more details. A reference solution is also provided for validation. Note that runs using GPU-acceleration require having an NVIDIA GPU with compatible drivers installed on the system.

6 Artifact Availability Statement

Summary of the Experiments Reported

Timings were performed on the various mini-app versions on the CPU and GPU using singularity containers. These versions represented different levels of replacing OpenACC and OpenMP directives with do concurrent loops. Each version was tested with the compilers gfortran, nvfortran, and ifort. The compilers were loaded in a singularity container, and the codes were executed through these singularity containers. For each code and compiler version, 10 runs were carried out in order to get an average run time and standard deviation.

Artifact Availability

Software Artifact Availability: All author-created software artifacts are maintained in a public repository under an OSI-approved license.

Hardware Artifact Availability: There are no author-created hardware artifacts.

Data Artifact Availability: All author-created data artifacts are maintained in a public repository under an OSI-approved license.

Proprietary Artifacts: No author-created artifacts are proprietary.

List of URLs and/or DOIs where artifacts are available:

10.5281/zenodo.5253520
http://www.predsci.com/papers/dc

Rights and permissions

Reprints and Permissions

Copyright information

About this paper

Cite this paper

Stulajter, M.M., Caplan, R.M., Linker, J.A. (2022). Can Fortran's 'do concurrent' Replace Directives for Accelerated Computing?. In: Bhalachandra, S., Daley, C., Melesse Vergara, V. (eds) Accelerator Programming Using Directives. WACCPD 2021. Lecture Notes in Computer Science(), vol 13194. Springer, Cham. https://doi.org/10.1007/978-3-030-97759-7_1

.RIS .ENW .BIB

DOI

https://doi.org/10.1007/978-3-030-97759-7_1

Published	Publisher Name	Print ISBN
15 May 2022	Springer, Cham	978-3-030-97758-0

Online ISBN	eBook Packages
978-3-030-97759-7	Computer Science
	Computer Science (R0)