Can Fortran’s `do concurrent’ replace directives for accelerated computing?

OpenACC BoF

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

www.predsci.com
Introduction

- Directives are popular for parallelization on CPUs & GPUs

**PROS**
- Easier to write than low-level APIs
- Performance can be similar to low-level APIs
- Portability
- Minimal code interference

**CONS**
- Not always supported
- Spec more fluid than language, so may need re-writes
- Can make code harder to read (e.g. deep copy, device type optimizations, etc.)

- Standard languages have begun to add features that compilers can use to parallelize code without directives:
  - C++17’s Standard Parallel Algorithms and Fortran’s `do concurrent`

**GOAL:** Test the current status of being able to replace directives with `do concurrent` for accelerated computing
Integrates the spherical surface heat equation on a logically rectangular non-uniform grid with a finite difference scheme which includes:

- Vector/array operations
- Stencil operations
- Reduction operations

Does NOT have:
- MPI (CUDA/RoCm-aware)
- Atomics
- Multi-GPU
- Derived types
- Function calls in loops

8 million point real-world problem is used for testing
## Computational Environment

<table>
<thead>
<tr>
<th>Compiler Suite</th>
<th>Compiler</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNU Compiler Collection</td>
<td>gfortran</td>
<td>11.2</td>
</tr>
<tr>
<td>NVIDIA HPC SDK</td>
<td>nvfortran</td>
<td>21.7</td>
</tr>
<tr>
<td>Intel OneAPI HPC Toolkit</td>
<td>ifort (classic)</td>
<td>21.3</td>
</tr>
</tbody>
</table>

### CPU Specifications

- **CPU/GPU Model**: (2x) AMD EPYC 7742 (128 cores)
- **Peak Memory Bandwidth**: 381.4 GB/s
- **Clock Frequency (base/boost)**: 2.3/3.4 GHz
- **RAM**: 256 GB
- **Peak DP FLOPs**: 7.0 TFLOPs

### GPU Specifications

- **GPU Type**: NVIDIA A100 SXM4
- **Memory Bandwidth**: 1555 GB/s
- **Clock Frequency**: 1.1/1.4 GHz
- **RAM**: 40 GB
- **TFLOPs**: 9.8 TFLOPs
• All CPU runs have similar performance for all compilers
• \texttt{nvfortran} runs faster on GPU than gfortran
Implementation: Fortran’s `do concurrent`

- Introduced in Fortran 2008
- Indicates loop can be run with out-of-order execution
- Can be used as hint to the compiler that loop may be parallelizable
- Current specification has no support for reductions, atomics, device selection, conditionals, etc.

**Code 1** Nested do loops with OpenMP/ACC directives

```fortran
!$omp parallel do collapse(2) default(shared)
!$acc parallel loop collapse(2) default(present)
  do i=1,N
    do j=1,M
      Computation
    enddo
  enddo
!$acc end parallel loop
!$omp end parallel do
```

**Code 2** Nested do loops as a do concurrent loop

```fortran
do concurrent (i=1:N, j=1:M)
  Computation
enddo
```
Experimental version represents "ideal" situation of having no directives
• Utilized `-03 and -march=<ARCH>` for all compilers

• **gfortran**:
  • CPU: `-fopenacc and/or -ftree-parallelize-loops=<N>`
  • GPU: `-fopenacc -foffload=nvptx-none -fopenacc-dim=::128`

• **nvfortran**:
  • CPU: `-mp or -acc=multicore`
  • GPU: `-stdpar=gpu and/or -acc=gpu -gpu=cc<XY>,cuda<X>.<Y>`
  • Note: managed memory is enabled by default when using `stdpar`! (can turn off with `-gpu=nomanaged`)

• **ifort**:
  • CPU: `-fopenmp`
  • GPU: No support for NVIDIA GPUs
## Results: nvfortran

### GPU

<table>
<thead>
<tr>
<th>Code</th>
<th>Compiler flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>-acc=gpu -gpu=cc80,cuda11.4</td>
</tr>
<tr>
<td>New</td>
<td>-acc=gpu -stdpar=gpu -gpu=cc80,cuda11.4</td>
</tr>
<tr>
<td>Experimental</td>
<td>-stdpar=gpu -gpu=cc80,cuda11.4</td>
</tr>
</tbody>
</table>

### CPU

<table>
<thead>
<tr>
<th>Code</th>
<th>Compiler flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial</td>
<td>-acc=multicore</td>
</tr>
<tr>
<td>Original</td>
<td>-acc=multicore -stdpar=multicore</td>
</tr>
<tr>
<td>New</td>
<td>-acc=multicore -stdpar=multicore</td>
</tr>
<tr>
<td>Experimental</td>
<td>-stdpar=multicore</td>
</tr>
</tbody>
</table>

### Wall Clock Time (seconds)

- **OpenACC A100 40GB**
  - Original: 35.1
  - New: 35.7
  - Experimental: 35.6

- **Serial 1-core**
  - Original (OpenACC): 224.2
  - New (DC+OpenACC): 219.6
  - Experimental (DC) 128-core: 219.2

- **128-cores**
  - Original (OpenACC): 1284.6

### Results: gfortran

<table>
<thead>
<tr>
<th>Code</th>
<th>Compiler flags</th>
<th>GPU</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>-fopenacc</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-foffload=nvptx-none</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-fopenacc-dim=::128</td>
<td></td>
<td></td>
</tr>
<tr>
<td>New</td>
<td>No Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Experimental</td>
<td>No Support</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**GNU gfortran (GPU)**

- **Original** (OpenACC): 49.5 seconds
- **New** (DC+OpenACC): Not Supported
- **Experimental** (DC): Not Supported

**GNU gfortran (CPU)**

- **Serial** 1-core: 1308.8 seconds
- **Original** (OpenMP) 128-cores: 191.9 seconds
- **New** (DC+OpenMP) 128-cores: 212.6 seconds
- **Experimental** (DC) 128-core: 236.3 seconds
## Results: ifort

### GPU
<table>
<thead>
<tr>
<th>Code</th>
<th>Compiler flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>No Support</td>
</tr>
<tr>
<td>New</td>
<td>No Support</td>
</tr>
<tr>
<td>Experimental</td>
<td>No Support</td>
</tr>
</tbody>
</table>

### CPU
<table>
<thead>
<tr>
<th>Code</th>
<th>Compiler flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial</td>
<td></td>
</tr>
<tr>
<td>Original</td>
<td>-fopenmp</td>
</tr>
<tr>
<td>New</td>
<td>-fopenmp</td>
</tr>
<tr>
<td>Experimental</td>
<td>Incorrect Results</td>
</tr>
</tbody>
</table>

![Bar chart showing wall clock time (seconds) for serial, OpenMP, and DC+OpenMP configurations.](chart.png)

- **Serial 1-core**: 1318.6 seconds
- **Original (OpenMP) 128-cores**: 194.9 seconds
- **New (DC+OpenMP) 128-cores**: 178.3 seconds
- **Experimental (DC)**: Incorrect Result
Discussion

- **Compatibility (GPU):**
  - Currently only nvfortran has do concurrent support for GPUs
  - Using do concurrent loses gfortran GPU support for now…
    … but may gain Intel GPU support with planned update to ifort
  - Relying on unified memory is possible performance loss (but not here)

- **Portability (CPU):**
  - CPU multi-core parallelization was retained (except for Experimental code on ifort)
  - nvfortran and ifort have direct support of do concurrent on CPUs, while gfortran relies on auto-parallelization
  - Implicit reductions with do concurrent not supported everywhere
• **Performance:**
  • Comparable performance between stdpar and directives on CPUs/GPUs

• **Summary:**
  • `do concurrent` allows cleaner code and increases “future-proofiness”
  • `nvfortran` allowed us to eliminate all directives!
  • … but using a combination of directives and `do concurrent` yields better cross compiler/hardware compatibility
So? Can Fortran's do concurrent replace directives for accelerated computing?

**YES! ...**

... with (1) `nvfortran`, (2) NVIDIA GPUs, (3) for some codes (like ours)


DOI: [10.5281/zenodo.5253520](https://doi.org/10.5281/zenodo.5253520)

Next step:

[github.com/predsci/POT3D](https://github.com/predsci/POT3D)

(spec.org/hpc2021)

Multi-GPU with MPI (CUDA/RoCm-aware), atomics, static arrays

It works! (with some !$acc)