MPAS
Model for Prediction Across Scales

Obtaining and Compiling MPAS-A

MPAS
Model for Prediction Across Scales

How do I get the code?

When getting started with a new modeling system, a reasonable first question to ask is:

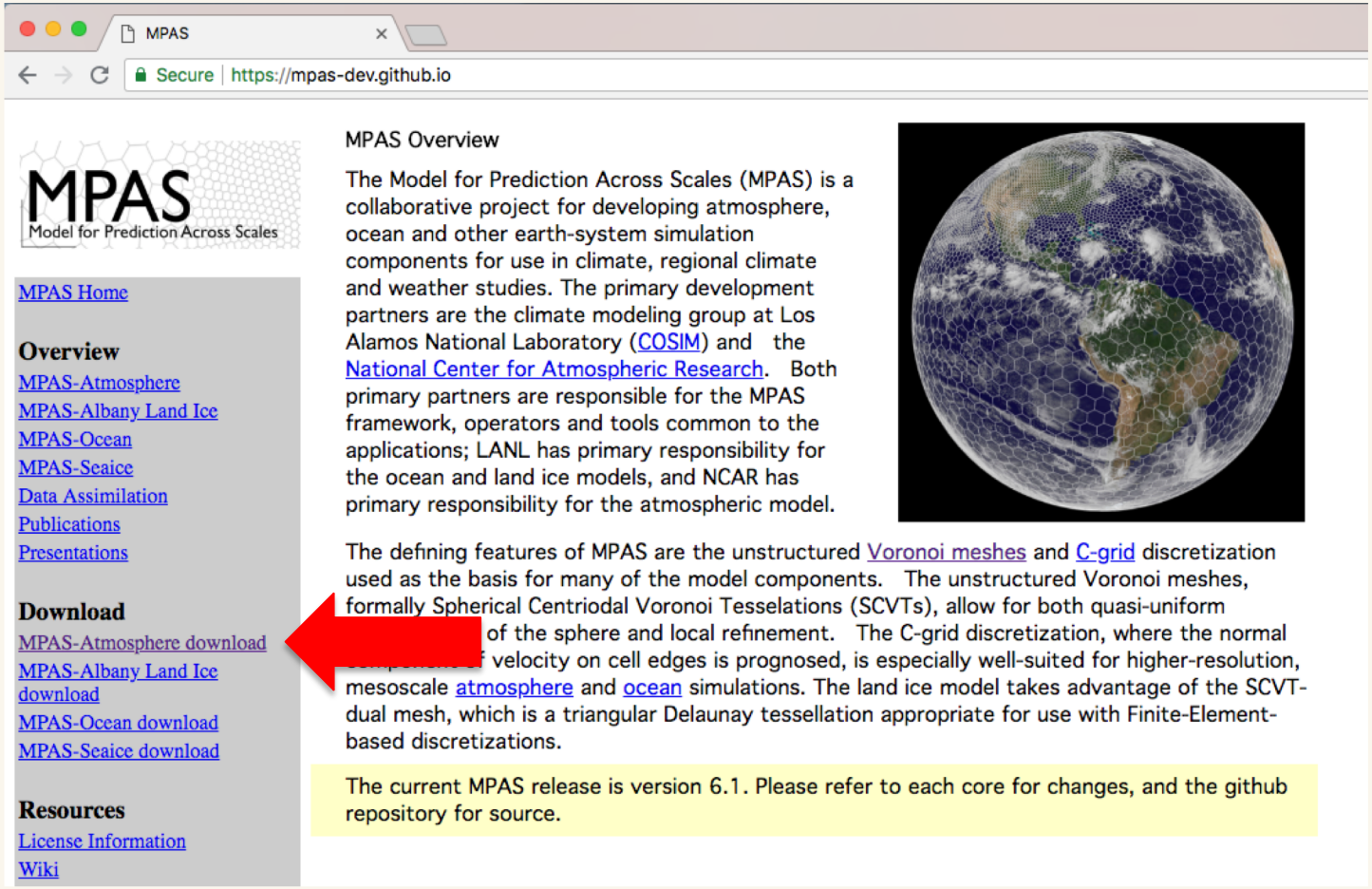
How do I get a copy of the source code?

1) The "traditional", but not necessarily encouraged, method
   - Download a .tar.gz or .zip file

2) The preferred method
   - Make a *clone* of the MPAS-Model repository

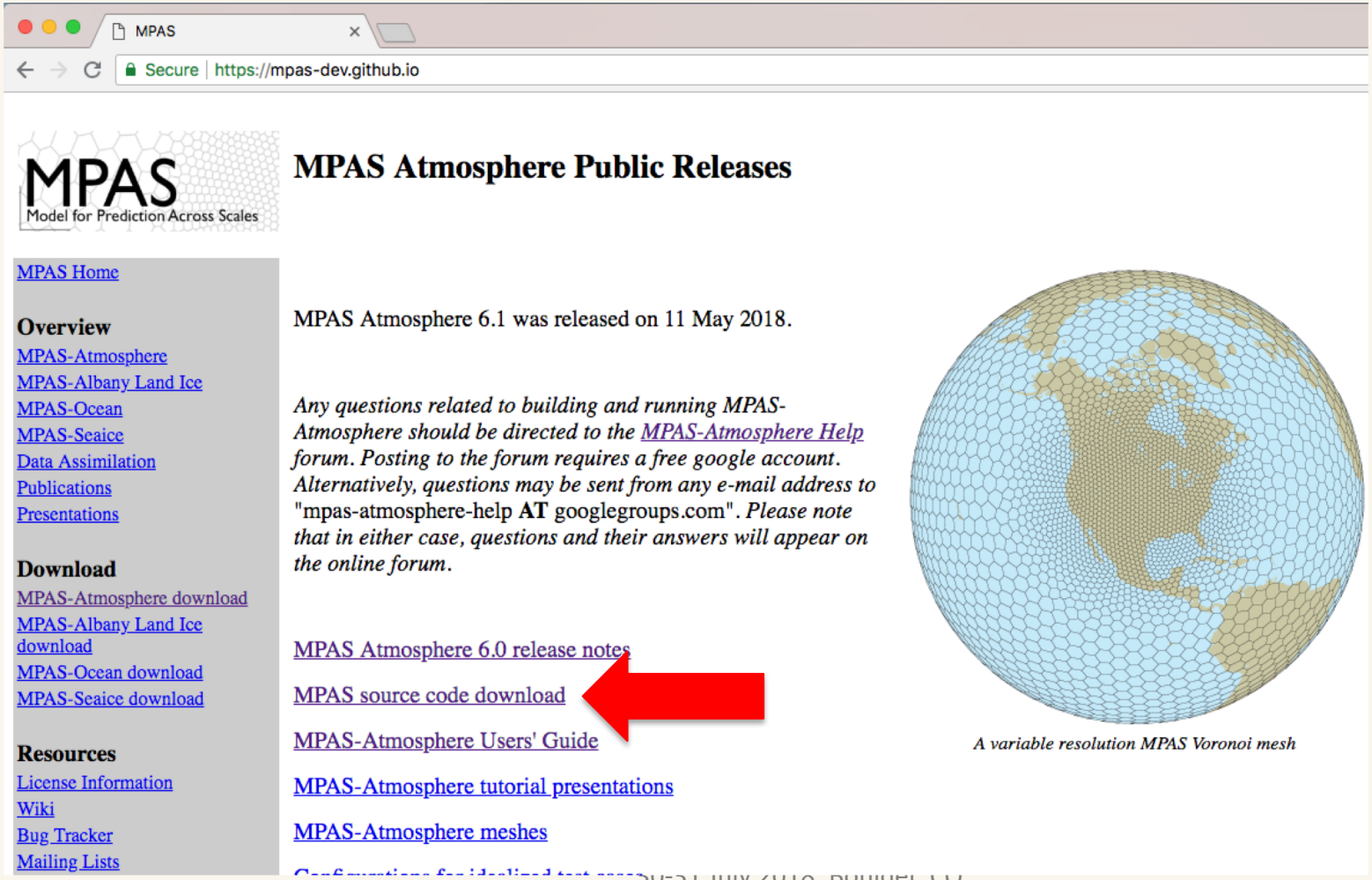
The ”less preferred” method of obtaining code

One can navigate to a download link from the MPAS homepage at https://mpas-dev.github.io/



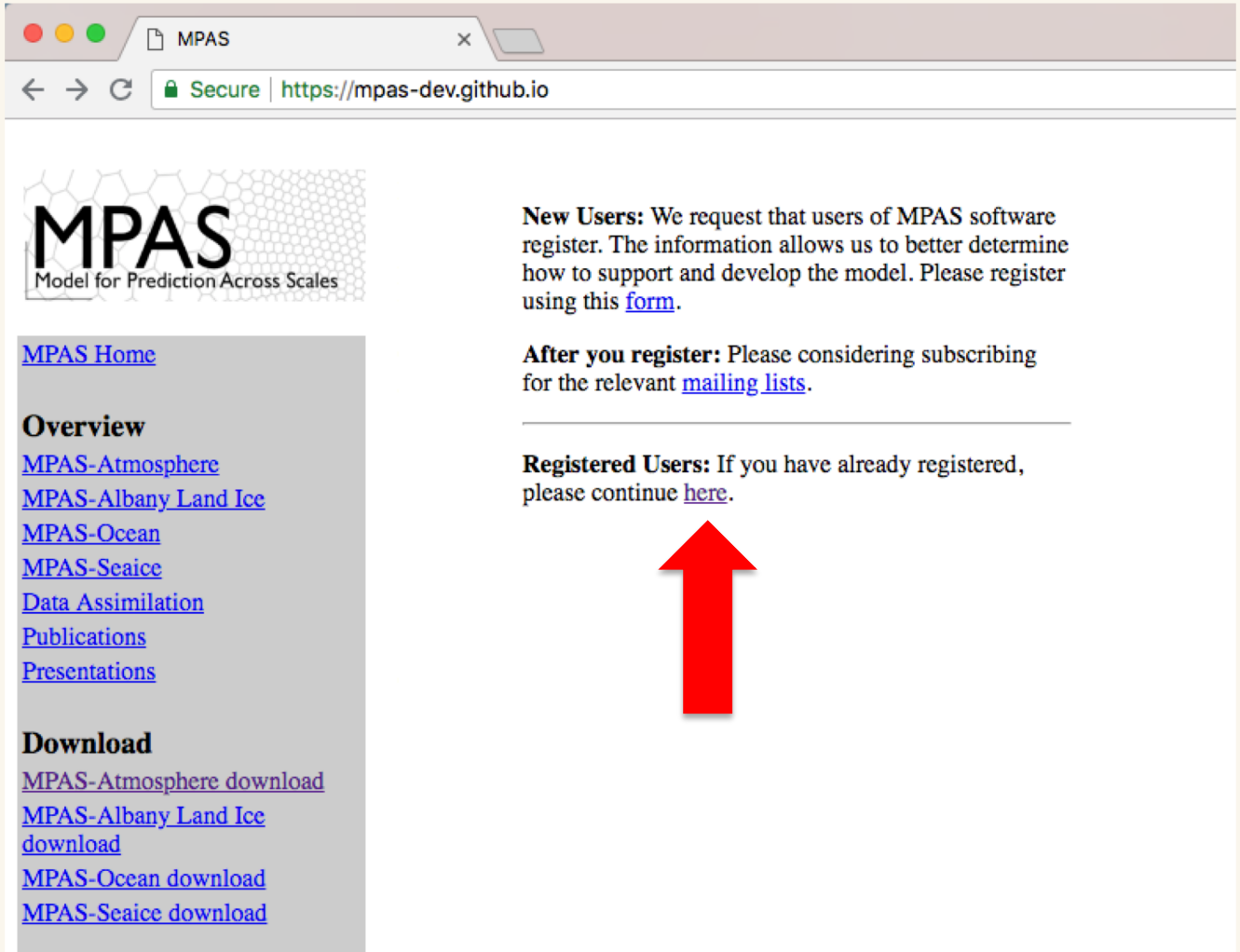
The ”less preferred” method of obtaining code

One can navigate to a download link from the MPAS homepage at https://mpas-dev.github.io/



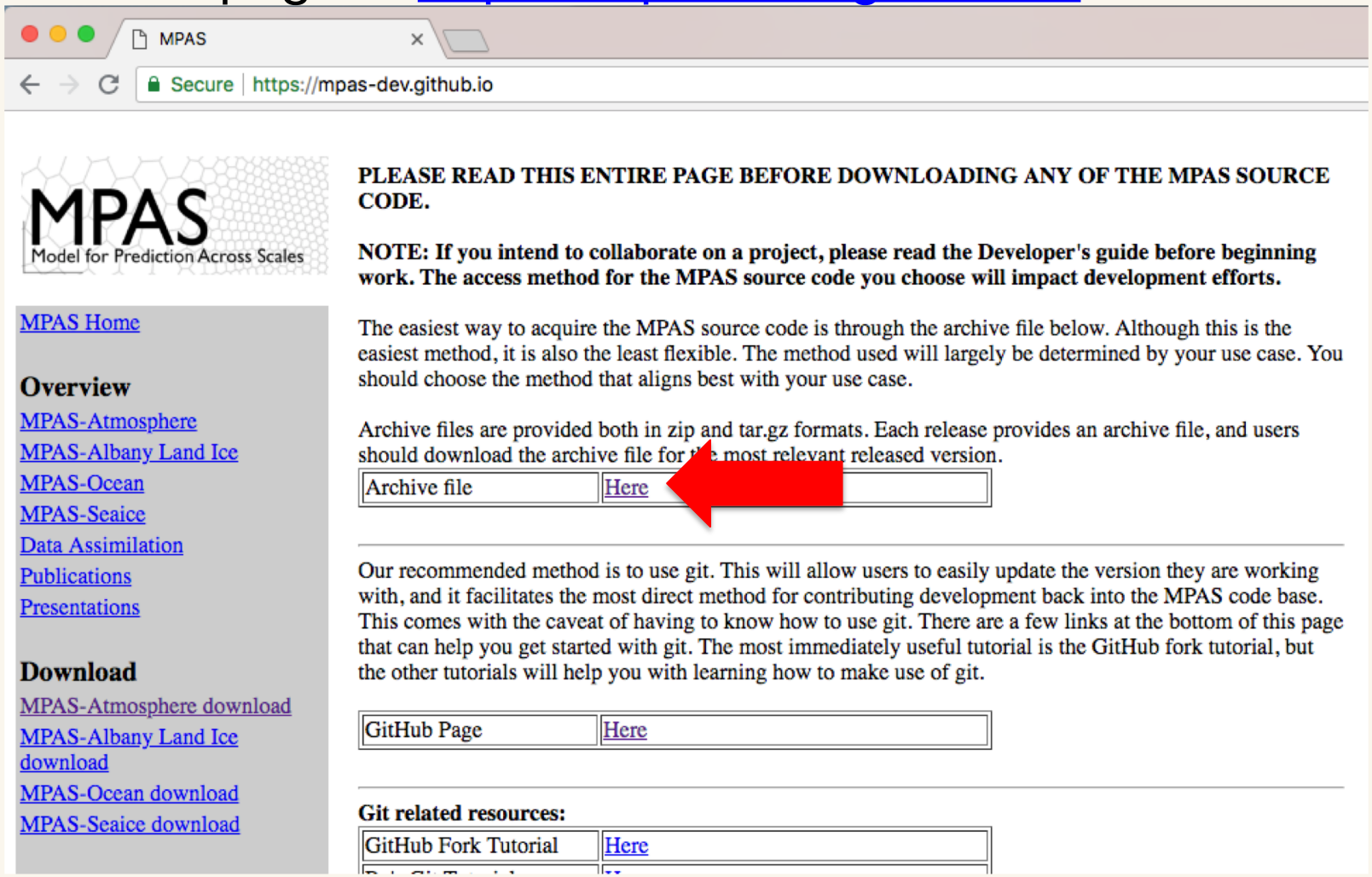
The ”less preferred” method of obtaining code

One can navigate to a download link from the MPAS homepage at https://mpas-dev.github.io/



The ”less preferred” method of obtaining code

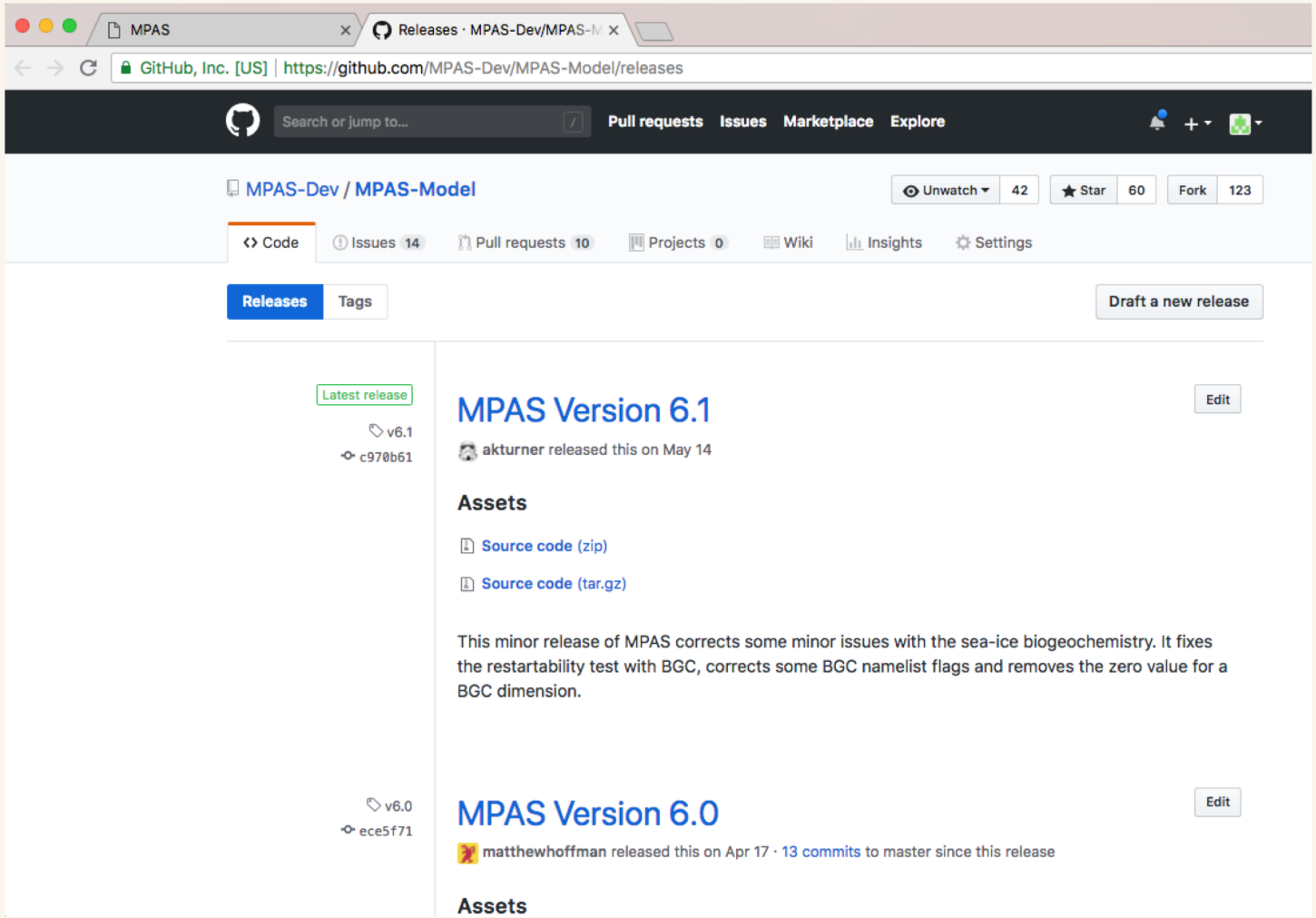
One can navigate to a download link from the MPAS homepage at https://mpas-dev.github.io/



MPAS
Model for Prediction Across Scales

The ”less preferred” method of obtaining code

One can navigate to a download link from the MPAS homepage at https://mpas-dev.github.io/



The ”less preferred” method of obtaining code

Downloading a .tar.gz file of a particular release of the MPAS code certainly works, but it has several disadvantages:

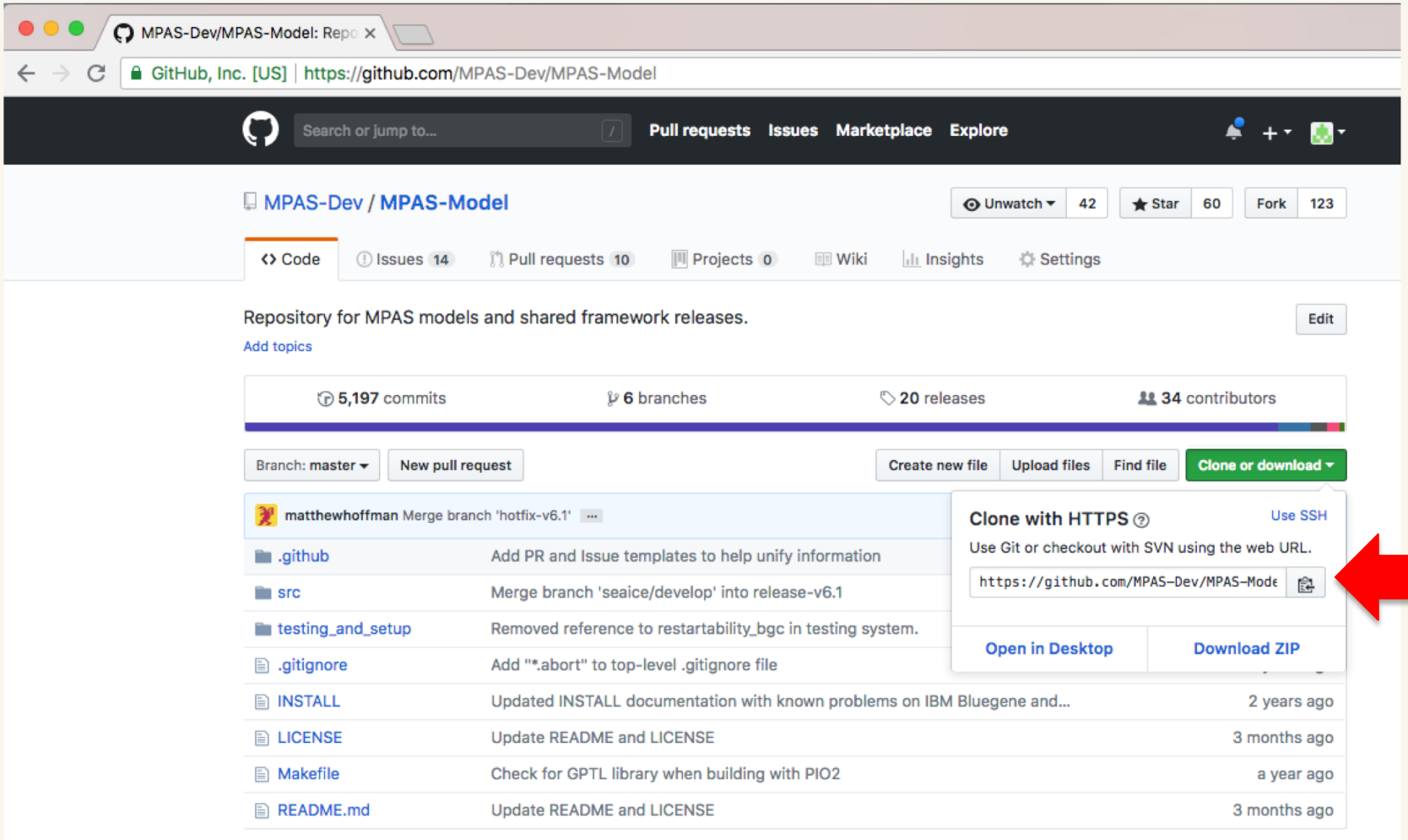
1. You’ll only obtain a specific release of the code
2. There’s no direct way to see the history of changes to parts of the code
3. There’s no easy route to updating to a newer release while preserving your local code modifications
4. It’s more difficult to see what local modifications have been made to the code
5. It’s more difficult to contribute improvements and fixes back to MPAS development

MPAS
Model for Prediction Across Scales

The preferred method of obtaining code

A much better option is to *clone* the MPAS-Model repository

- The repository URL can be found from the MPAS GitHub page at https://github.com/MPAS-Dev/MPAS-Model

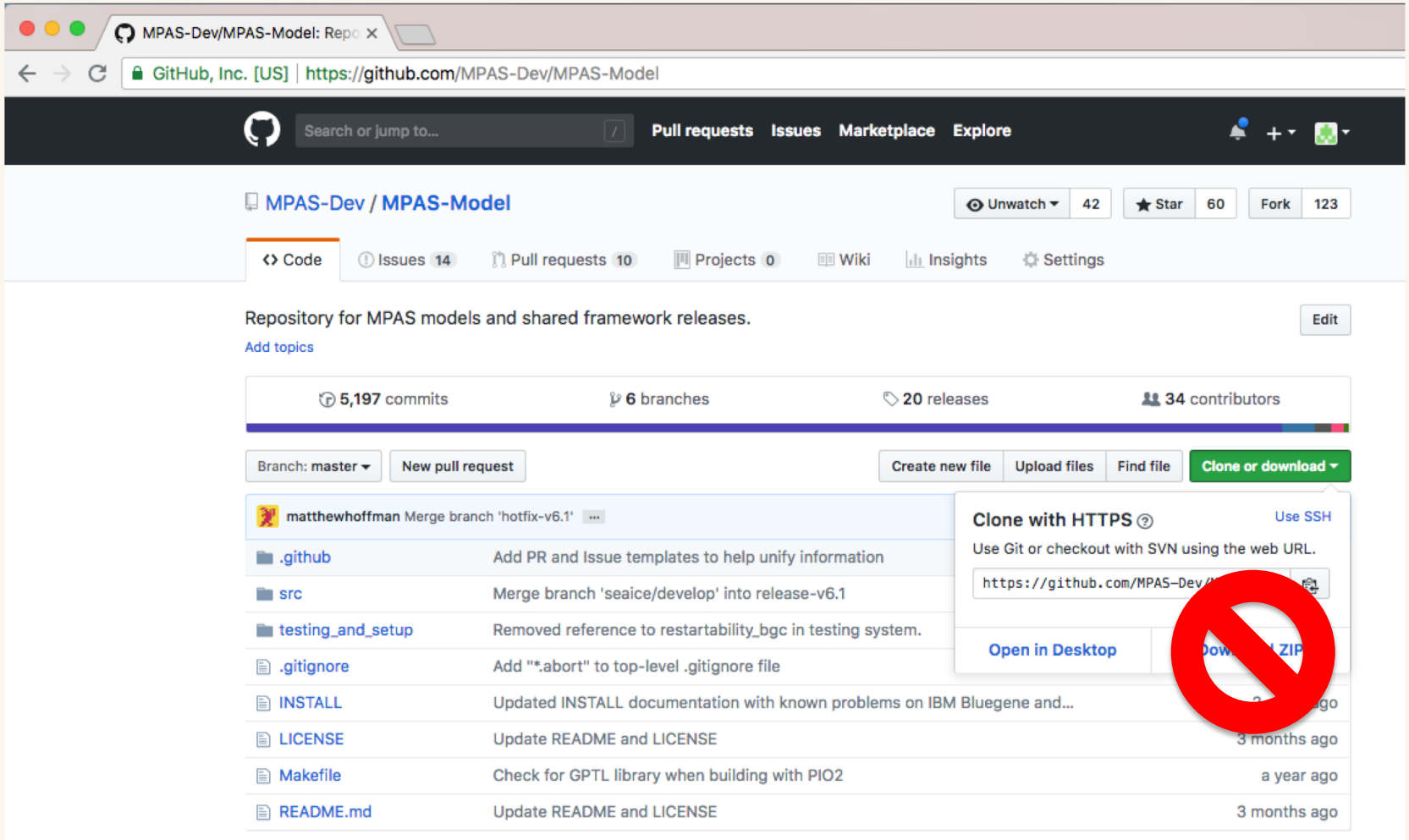


MPAS
Model for Prediction Across Scales

The preferred method of obtaining code

A much better option is to *clone* the MPAS-Model repository

- The repository URL can be found from the MPAS GitHub page at https://github.com/MPAS-Dev/MPAS-Model



The preferred method of obtaining code

From the command-line, the following should be sufficient:

```
git clone https://github.com/MPAS-Dev/MPAS-Model.git
```

Cloning the repository should take about 10 seconds or less…*

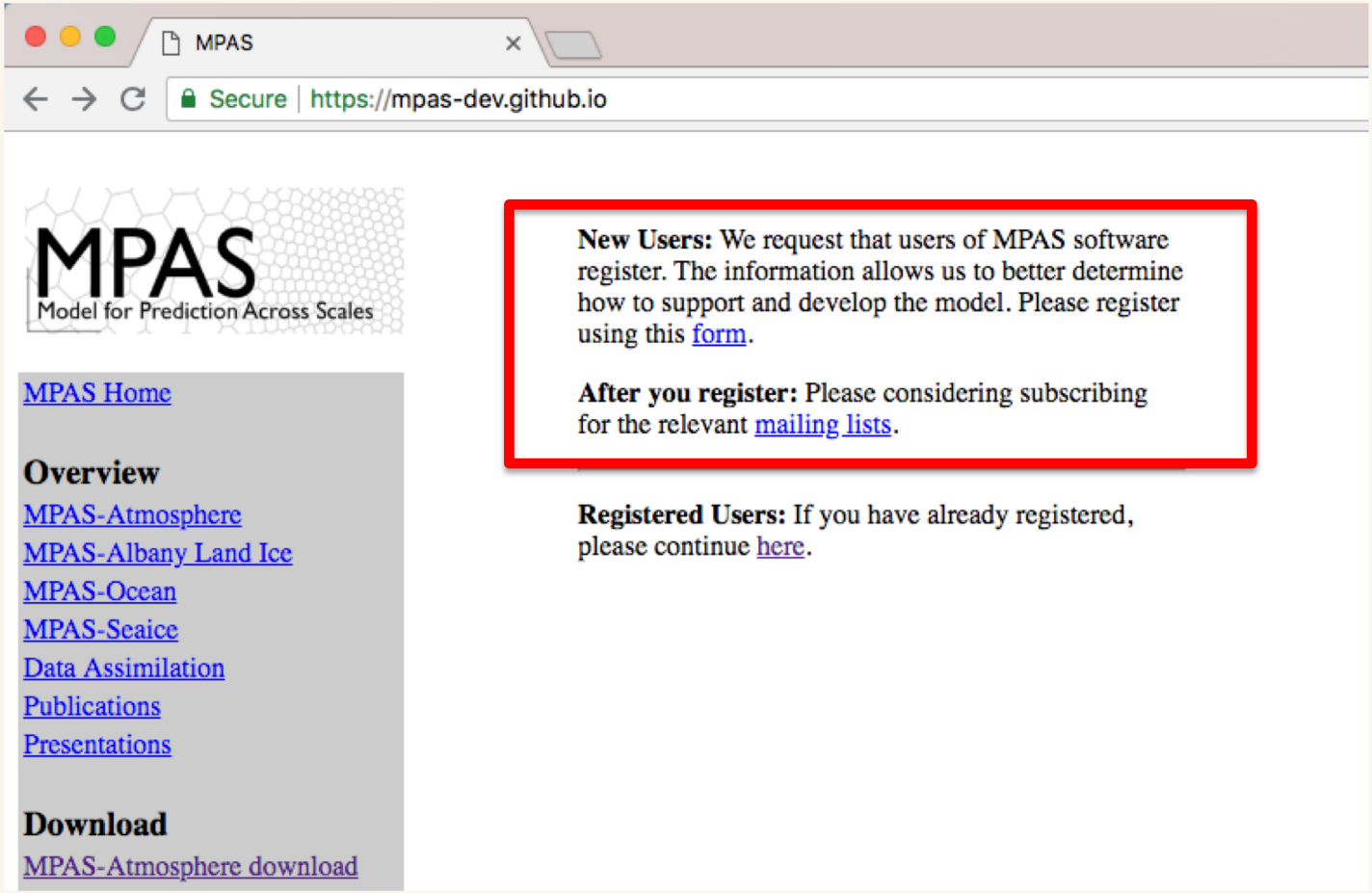
```
$ git clone https://github.com/MPAS-Dev/MPAS-Model.git
Cloning into 'MPAS-Model'...
remote: Counting objects: 38647, done.
remote: Total 38647 (delta 3), reused 3 (delta 3), pack-reused 38643
Receiving objects: 100% (38647/38647), 16.22 MiB | 11.25 MiB/s, done.
Resolving deltas: 100% (30112/30112), done.

$ ls -d MPAS-Model/
MPAS-Model/
```

*… except when GitHub is being hit by a DDoS attack.

After obtaining the code for the first time

You may also like to register as an MPAS user and join the MPAS-Atmosphere Users mailing list



Preliminary requirements

In order to compile MPAS and its required libraries, working C and Fortran compilers are necessary

- The Fortran compiler should be recent enough to support the ISO_C_BINDING module from the Fortran 2003 standard and procedure pointer components of derived types
- Most versions of common compilers from the last couple of years should be fine

Building MPAS requires *at least* the following libraries:

- Any implementation of MPI-2, e.g., MPICH, MVAPICH, OpenMPI
  - Ensure that `mpif90` and `mpicc` commands are in your path
- Parallel-netCDF (http://trac.mcs.anl.gov/projects/parallel-netcdf/)
  - Set `PNETCDF` environment variable to base installation directory
- PIO (https://github.com/NCAR/ParallelIO/)
  - Set `PIO` environment variable to base installation directory

Preliminary requirements: the easiest route

Assuming Fortran and C compilers are available, and a working MPI installation is also available, installing Parallel-netCDF and PIO should take less than 10 minutes:

Parallel-netCDF 1.8.1

```
$ setenv CC gcc
$ setenv FC gfortran
$ setenv F77 gfortran
$ setenv MPICC mpicc
$ setenv MPIF90 mpif90
$ setenv MPIF77 mpif90
$ setenv PNETCDF /home/duda/pnetcdf
$ cd parallel-netcdf-1.8.1
$ ./configure \
  --prefix=$PNETCDF \
  --disable-cxx
$ make
$ make install
```

PIO 1.7.1

(Assuming environment variables from Parallel-NetCDF installation)

```
$ setenv MPIFC mpif90
$ setenv PNETCDF_PATH $PNETCDF
$ setenv PIO /home/duda/pio
$ cd pio1_7_1/pio
$ ./configure \
  --prefix=$PIO \
  --disable-netcdf \
  --disable-mpiio
$ make
$ make install
```

Which versions of PIO should I try?

The PIO library is undergoing rapid development, and many different versions of the library are available; *which versions are supported and recommended?*

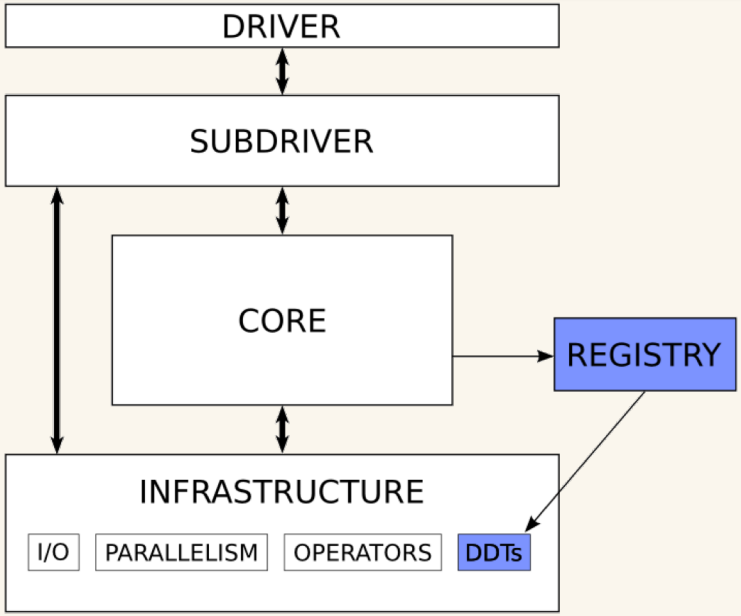
1) For ease of installation, try PIO 1.7.1

- Can be installed using only standard 'configure' and 'make' tools
- Supports NetCDF-3 and Parallel-netCDF I/O
- Download: https://github.com/NCAR/ParallelIO/releases/tag/pio1_7_1

2) If netCDF-4 I/O is needed or desirable, try the latest PIO release

- Requires recent versions of *'cmake'*, plus standard 'make'
- Supports netCDF-3, netCDF-4 (in parallel via PHDF5), and Parallel-netCDF I/O
- Download: https://github.com/NCAR/ParallelIO/
- See iolib_installation.sh at http://www2.mmm.ucar.edu/people/duda/files/mpas/sources/

Model Organization

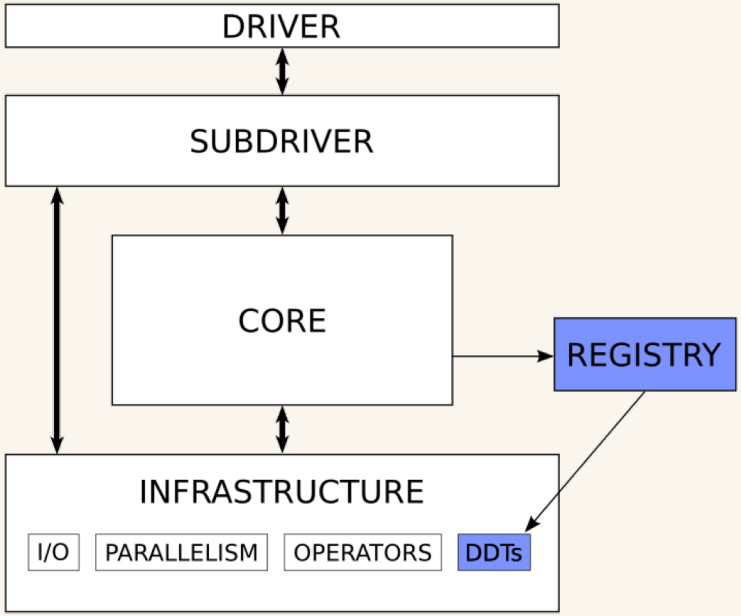


Checking out the MPAS code provides all MPAS models, not just MPAS-Atmosphere

- All models share a common set of infrastructure modules
- Each MPAS model is implemented as a "core" that lives in its own directory
- User must select which "core" to compile
- Each "core" is associated with a source code subdirectory under src/ and has a Registry file (similar to WRF)

```
MPAS-Model/
    └──> src/
            ├──> core_atmosphere/
            ├──> core_init_atmosphere/
            ├──> core_landice/
            ├──> core_ocean/
            ├──> driver/
            ├──> external/
            ├──> framework/
            └──> operators/
```

Model Organization



Checking out the MPAS code provides all MPAS models, not just MPAS-Atmosphere

- All models share a common set of infrastructure modules
- Each MPAS model is implemented as a "core" that lives in its own directory
- User must select which "core" to compile
- Each "core" is associated with a source code subdirectory under src/ and has a Registry file (similar to WRF)

Running MPAS-Atmosphere involves two "cores":

- The **init_atmosphere** core is responsible for
  - Interpolating static fields to the mesh (similar to geogrid.exe)
  - Generating a vertical grid (similar to real.exe)
  - Horizontally and vertically interpolating meteorological data to the 3-d grid (similar to metgrid.exe and real.exe)
  - *Where to we get meteorological data? From ungrib.exe!*
- The **atmosphere** core is the model itself, the equivalent of wrf.exe

MPAS
Model for Prediction Across Scales

Compiling MPAS

There is no "configuration" step for MPAS, unlike, e.g., for the WRF model

- All build flags are either set in the top-level Makefile or on the command-line

General MPAS build command:

```
$ make target CORE=core <options>
```

target can be either

```
clean
```

or

```
xlf
gfortran
ifort
pgi
bluegene
... plus a few others...
```

For MPAS-Atmosphere, core may be

```
atmosphere
init_atmosphere
```

<options> can be zero or more of

```
DEBUG=true
AUTOCLEAN=true
PRECISION=single
OPENMP=true
USE_PIO2=true
```

MPAS
Model for Prediction Across Scales

Compiling MPAS

There is no "configuration" step for MPAS, unlike, e.g., for the WRF model

- All build flags are either set in the top-level Makefile or on the command-line

General MPAS build command:

```
$ make target CORE=core <options>
```

target can be either

```
clean
```

or

```
xlf
gfortran
ifort
pgi
bluegene
... plus a few others...
```

For MPAS-Atmosphere, core may be

```
atmosphere
init_atmosphere
```

<options> (

```
DEBUG=
AUTOCL
PRECISIO
OPENMP=true
USE_PIO2=true
```

If using the latest PIO2 library, be sure to set this option!

MPAS
Model for Prediction Across Scales

Compiling MPAS

Typical build of both the init_atmosphere and atmosphere cores involves:

$ make gfortran CORE=init_atmosphere *(build init_atmosphere_model)*

$ make clean CORE=atmosphere *(clean any infrastructure files used by both init_atmosphere and atmosphere)*

$ make gfortran CORE=atmosphere *(build atmosphere_model)*

By default, MPAS cores are built with double-precision reals

MPAS-Atmosphere can be built in single precision

- Add `PRECISION=single` to build commands for single-precision executables
- execution time ~35% less compared with double-precision
- output files approximately half as large
- Beginning with MPAS v3.0, it is possible to run the model in double precision while writing history files in single precision!

MPAS
Model for Prediction Across Scales

Cheyenne users: you're in luck!

If you're working on Cheyenne, there are just seven commands to obtain and build everything you need (assuming the default module setup):

```
module unload netcdf
module load pio
git clone https://github.com/MPAS-Dev/MPAS-Model.git
cd MPAS-Model
make ifort CORE=init_atmosphere PRECISION=single USE_PIO2=true
make clean CORE=atmosphere
make ifort CORE=atmosphere PRECISION=single USE_PIO2=true
```

MPAS
Model for Prediction Across Scales

Don't forget to check the Users' Guide!

Chapter 3 of the MPAS-Atmosphere Users' Guide provides more details on compiling I/O library prerequisites as well as the MPAS-A model itself

Chapter 3

Building MPAS

3.1 Prerequisites

To build MPAS, compatible C and Fortran compilers are required. Additionally, the MPAS software relies on the PIO parallel I/O library to read and write model fields, and the PIO library requires the standard NetCDF library as well as the Parallel-NetCDF library from Argonne National Laboratory. All libraries must be compiled with the same compilers that will be used to build MPAS. Section 3.2 summarizes the basic procedure of installing the required I/O libraries for MPAS.

In order for the MPAS makefiles to find the PIO, Parallel-NetCDF, and NetCDF include files and libraries, the environment variables `PIO`, `PNETCDF`, and `NETCDF` should be set to the root installation directories of the PIO, Parallel-NetCDF, and NetCDF installations, respectively.

An MPI installation such as MPICH or OpenMPI is also required, and there is no option to build a serial version of the MPAS executables. MPAS-Atmosphere v5.0 introduces the capability to use hybrid parallelism using MPI and OpenMP; however, the use of OpenMP *should be considered experimental* and generally does not offer any performance advantage. The primary reason for releasing a shared-memory capability is to make this code available to collaborators for future development.

3.2 Compiling I/O Libraries

IMPORTANT NOTE: *The instructions provided in this section for installing libraries have been successfully used by MPAS developers, but due to differences in library versions, compilers, and system configurations, it is recommended that users consult documentation provided by individual*