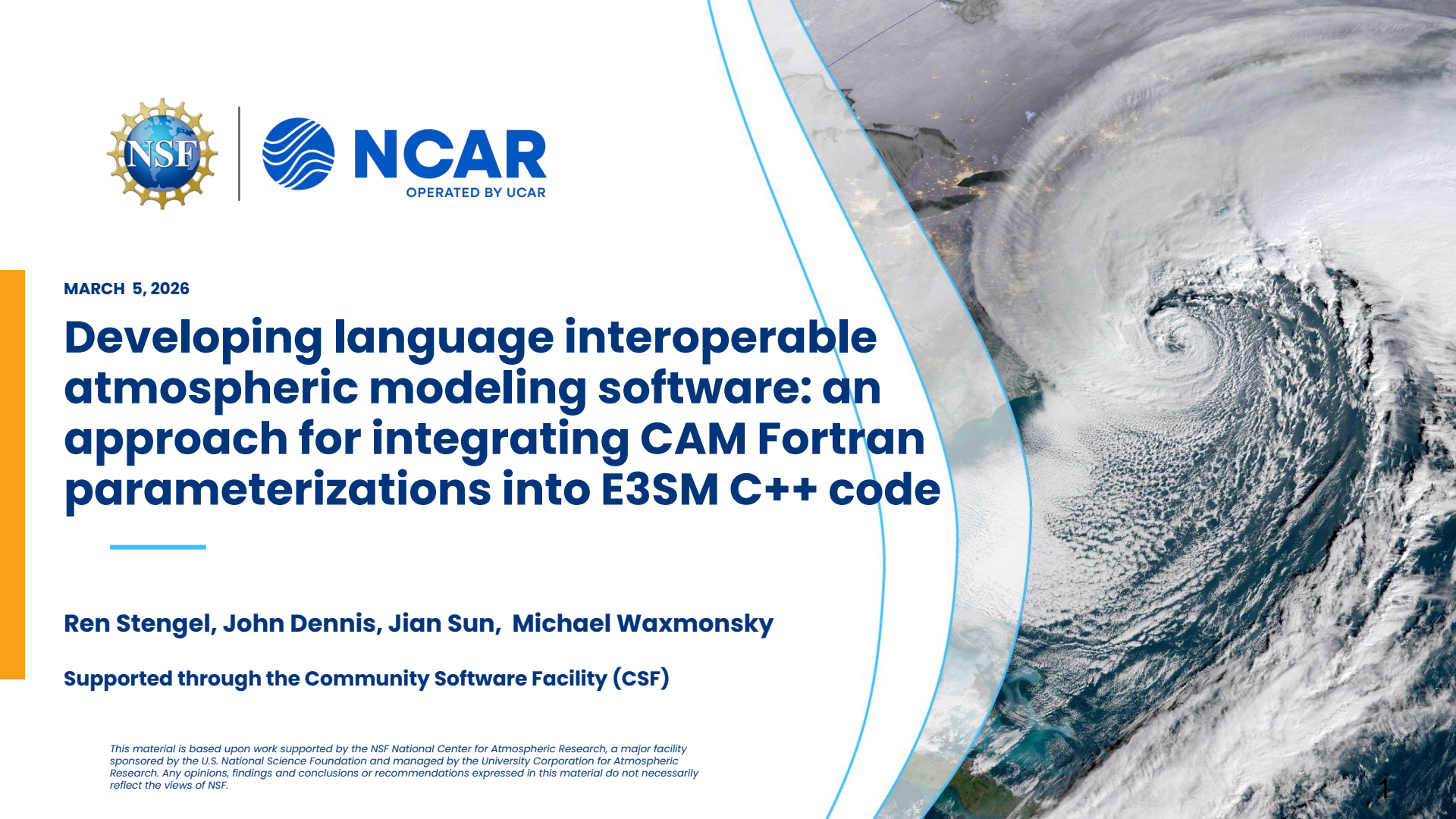


MARCH 5, 2026

Developing language interoperable atmospheric modeling software: an approach for integrating CAM Fortran parameterizations into E3SM C++ code

Ren Stengel, John Dennis, Jian Sun, Michael Waxmonsky

Supported through the Community Software Facility (CSF)

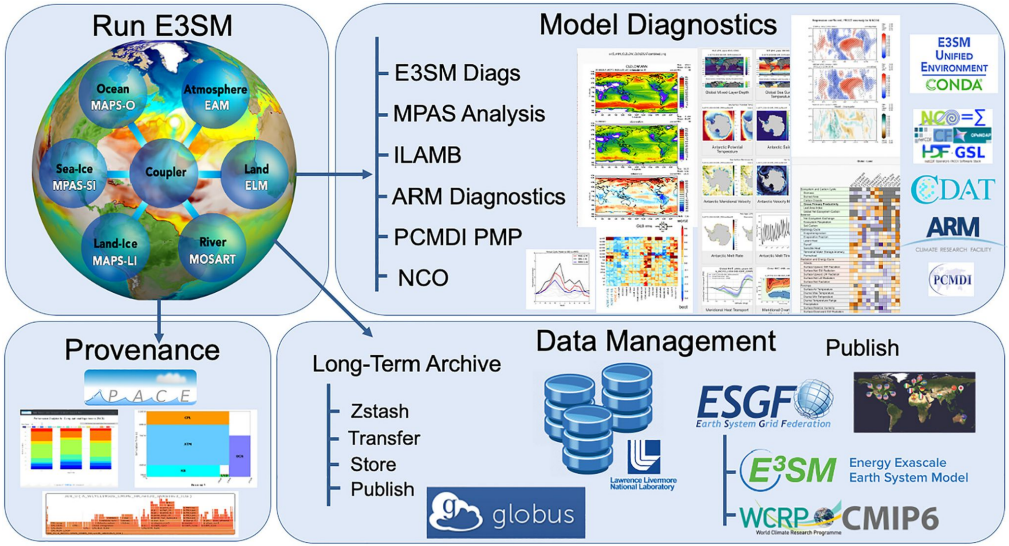
This material is based upon work supported by the NSF National Center for Atmospheric Research, a major facility sponsored by the U.S. National Science Foundation and managed by the University Corporation for Atmospheric Research. Any opinions, findings and conclusions or recommendations expressed in this material do not necessarily reflect the views of NSF.

Motivation

- Multiple ongoing/past projects that involve code modernization
  - CCPP: Interoperable physics written in Fortran
  - Earthworks: GPU-enablement through OpenACC directives
  - StormSPEED: Integration of performance-portable Kokkos/C++ dynamical core
  - Experimentation with JAX
- Code modernization efforts have different goals
  - Complexity containment
  - Physics package portability
  - GPU-enablement
- Observations:
  - Primary industry support is: Python and C++
  - Significantly less support for Fortran and even less for Fortran on GPUs
  - Graduate students/Postdocs (i.e. **future scientific leaders**) are using Python
- How would multi-language support impact community contribution?
  - StormSPEED: C++ dynamical core in Fortran application (CAM)
  - This project: Fortran physics in Kokkos/C++ application (EAMxx)

Energy Exascale Earth System Model (E3SM)

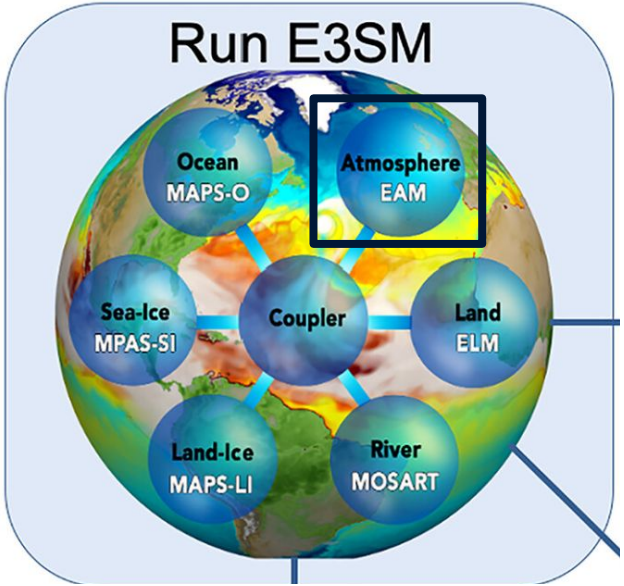
- E3SM is DOE's earth system modeling framework
- The goal is to provide a framework capable of exascale simulations



https://doi.org/10.1029/2019MS001821

EAMxx – atmospheric modeling component

- **Supports language interoperability for parameterizations and physics**
- Hardware portability and performance on GPUs with Kokkos



We will focus on the atmospheric component: EAMxx

https://doi.org/10.1029/2019MS001821

E3SM simulation description interface

- Uses the standard **`./xmlchange`** functionality:
  - Select COMPSET and grid
  - Set up compilers, machine environment, and batch job parameters
  - Simulated run time, time stepping, etc

```
./xmlchange RUN_TYPE="startup"
if [[ $COMPSET == *"F20TR"* ]]; then
   ./xmlchange RUN_STARTDATE='1850-01-01'
elif [[ $COMPSET == "FMTHIST" || $COMPSET == "FLTHIST" ]]; then
   ./xmlchange RUN_STARTDATE='2001-01-01'
else
   ./xmlchange RUN_STARTDATE='0001-01-01'
fi
./xmlchange RESUBMIT='0'
./xmlchange CONTINUE_RUN='FALSE'
./xmlchange STOP_N='2',STOP_OPTION='ndays'
./xmlchange JOB_WALLCLOCK_TIME='00:15:00'
./xmlchange JOB_QUEUE=$QUEUE_NAME
./xmlchange BUDGETS=TRUE

if [[ $DYCORE == "theta-l_kokkos" ]]; then
cat << EOF >> user_nl_elm
   check_finidat_year_consistency = .false.
   check_dynpft_consistency = .false.
   create_crop_landunit = .false.
EOF
fi
```

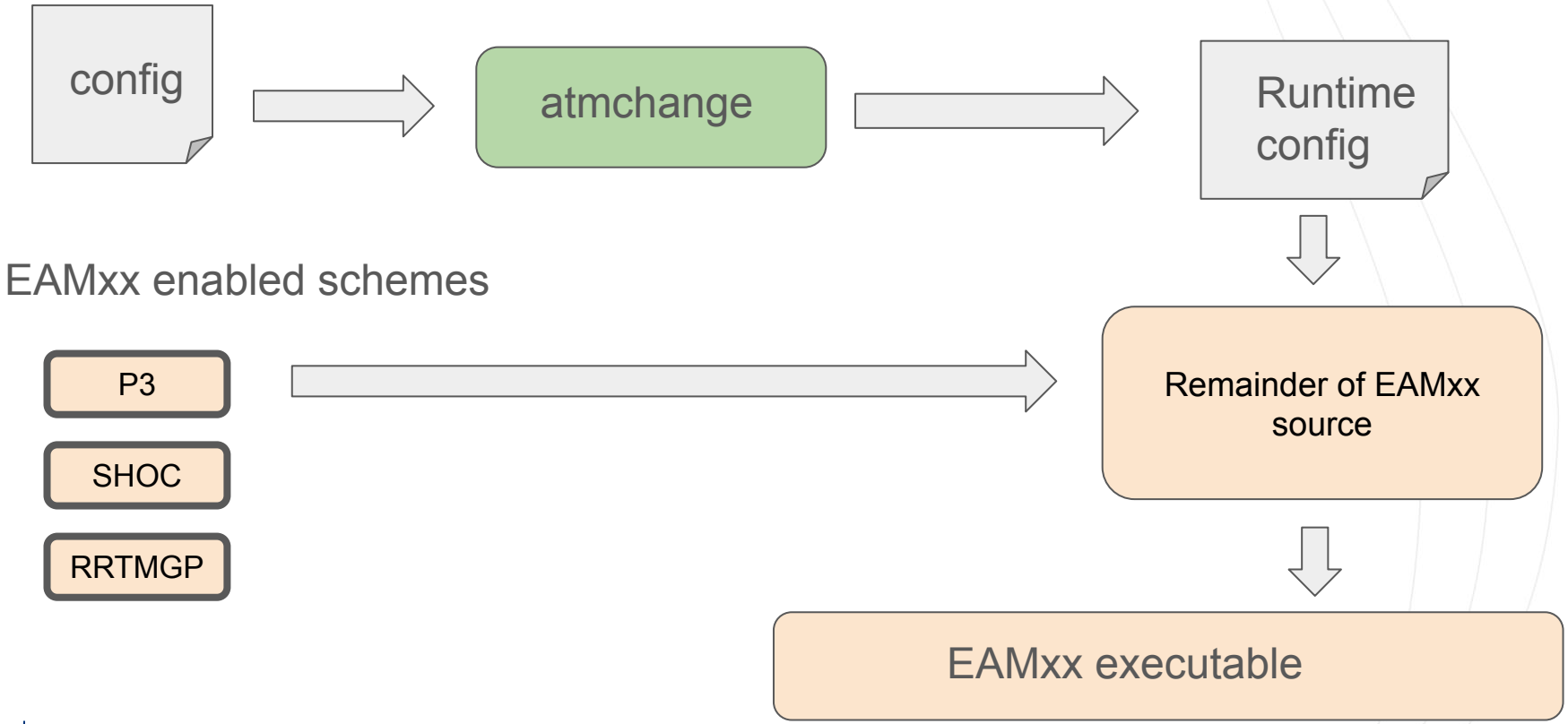
E3SM simulation description interface

- Uses the standard **`./xmlchange`** functionality:
  - Select COMPSET and grid
  - Set up compilers, machine environment, and batch job parameters
  - Simulated run time, time stepping, etc
- Uses **`./atmchange`** to change atmospheric process settings at runtime
- CCPP (used by CAM-SIMA) uses XML files to determine atmospheric process settings which then generates code to compile

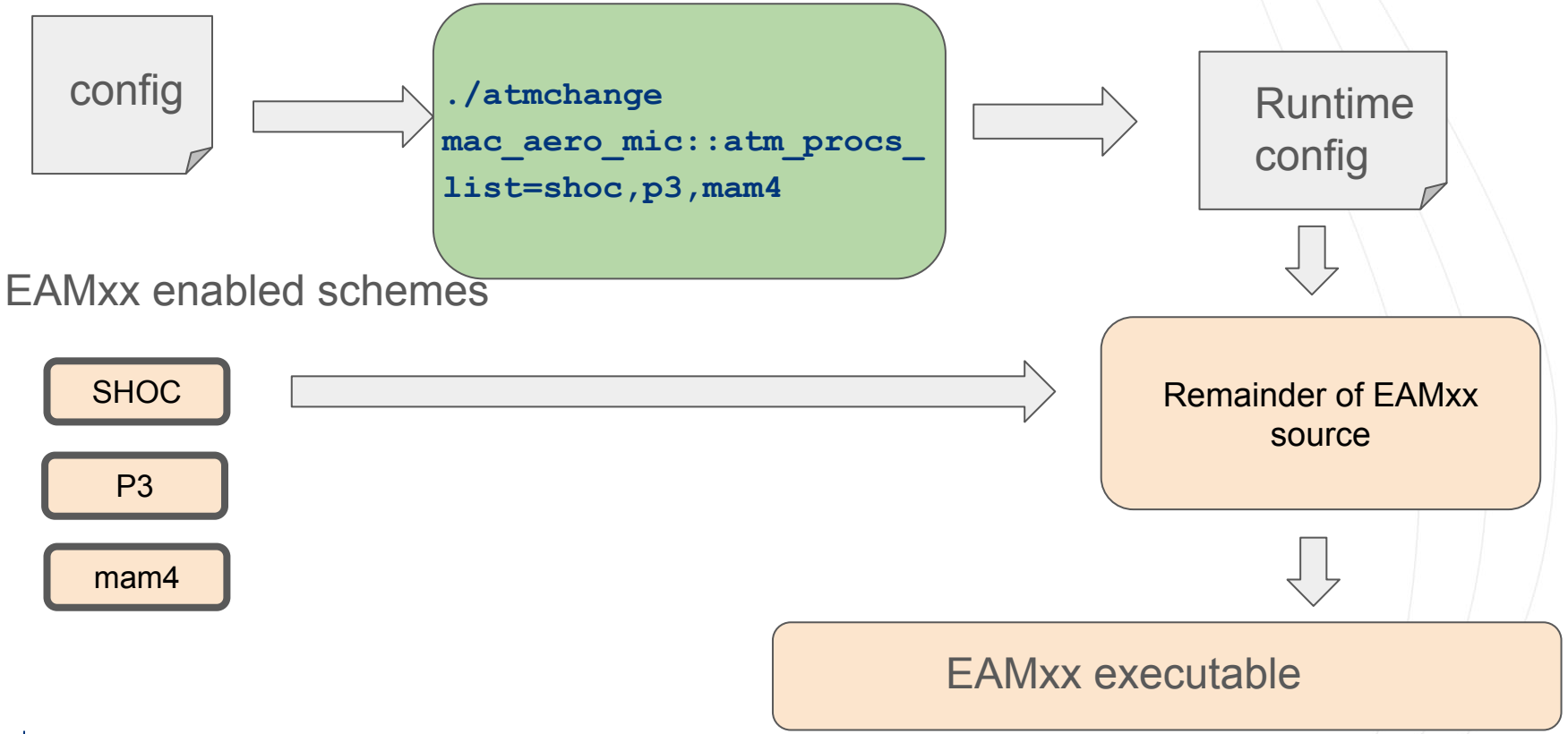
```
./xmlchange RUN_TYPE="startup"
if [[ $COMPSET == *"F20TR"* ]]; then
   ./xmlchange RUN_STARTDATE='1850-01-01'
elif [[ $COMPSET == "FMTHIST" || $COMPSET == "FLTHIST" ]]; then
   ./xmlchange RUN_STARTDATE='2001-01-01'
else
   ./xmlchange RUN_STARTDATE='0001-01-01'
fi
./xmlchange RESUBMIT='0'
./xmlchange CONTINUE_RUN='FALSE'
./xmlchange STOP_N='2',STOP_OPTION='ndays'
./xmlchange JOB_WALLCLOCK_TIME='00:15:00'
./xmlchange JOB_QUEUE=$QUEUE_NAME
./xmlchange BUDGETS=TRUE

if [[ $DYCORE == "theta-l_kokkos" ]]; then
cat << EOF >> user_nl_elm
   check_finidat_year_consistency = .false.
   check_dynpft_consistency = .false.
   create_crop_landunit = .false.
EOF
fi
```

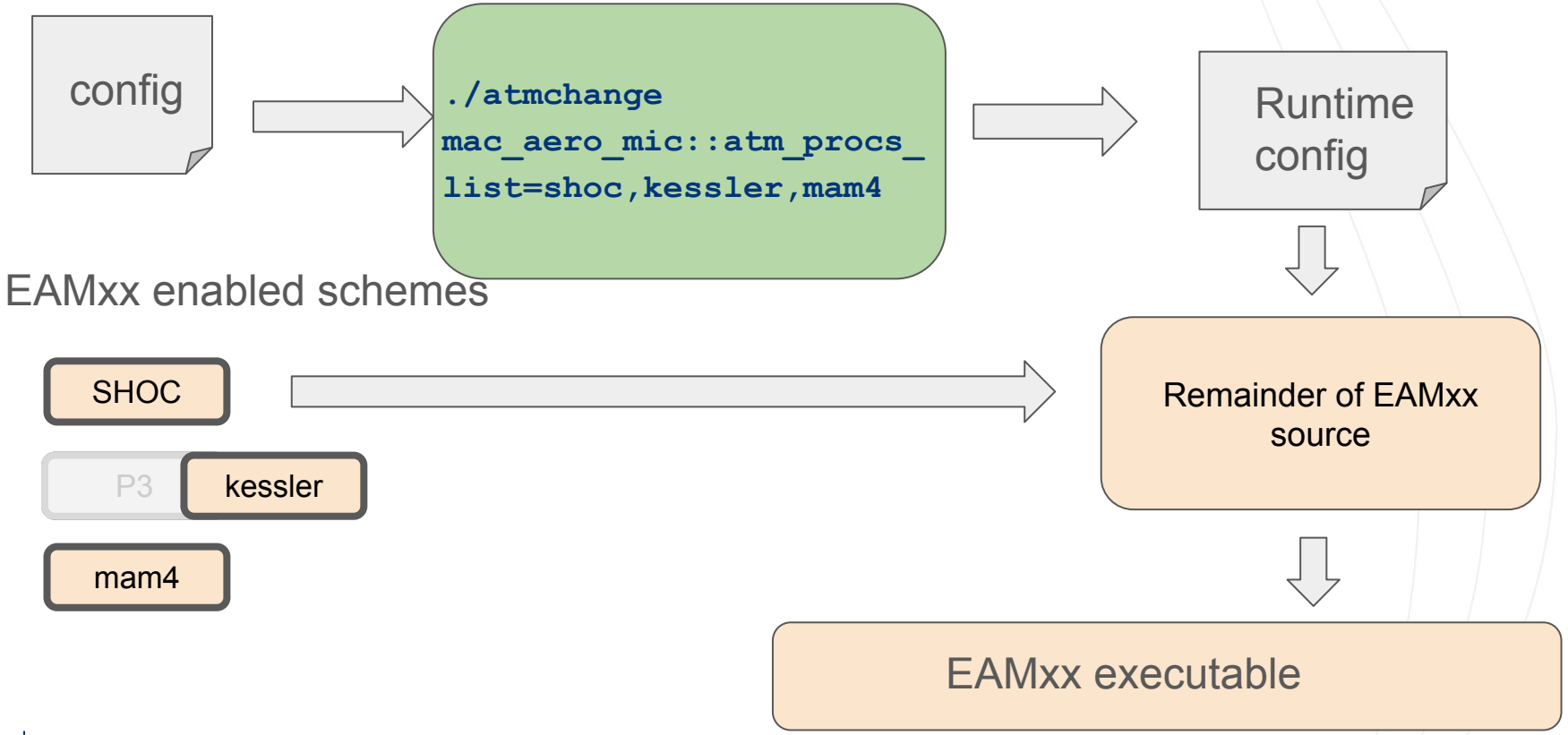
EAMxx change atmospheric processes



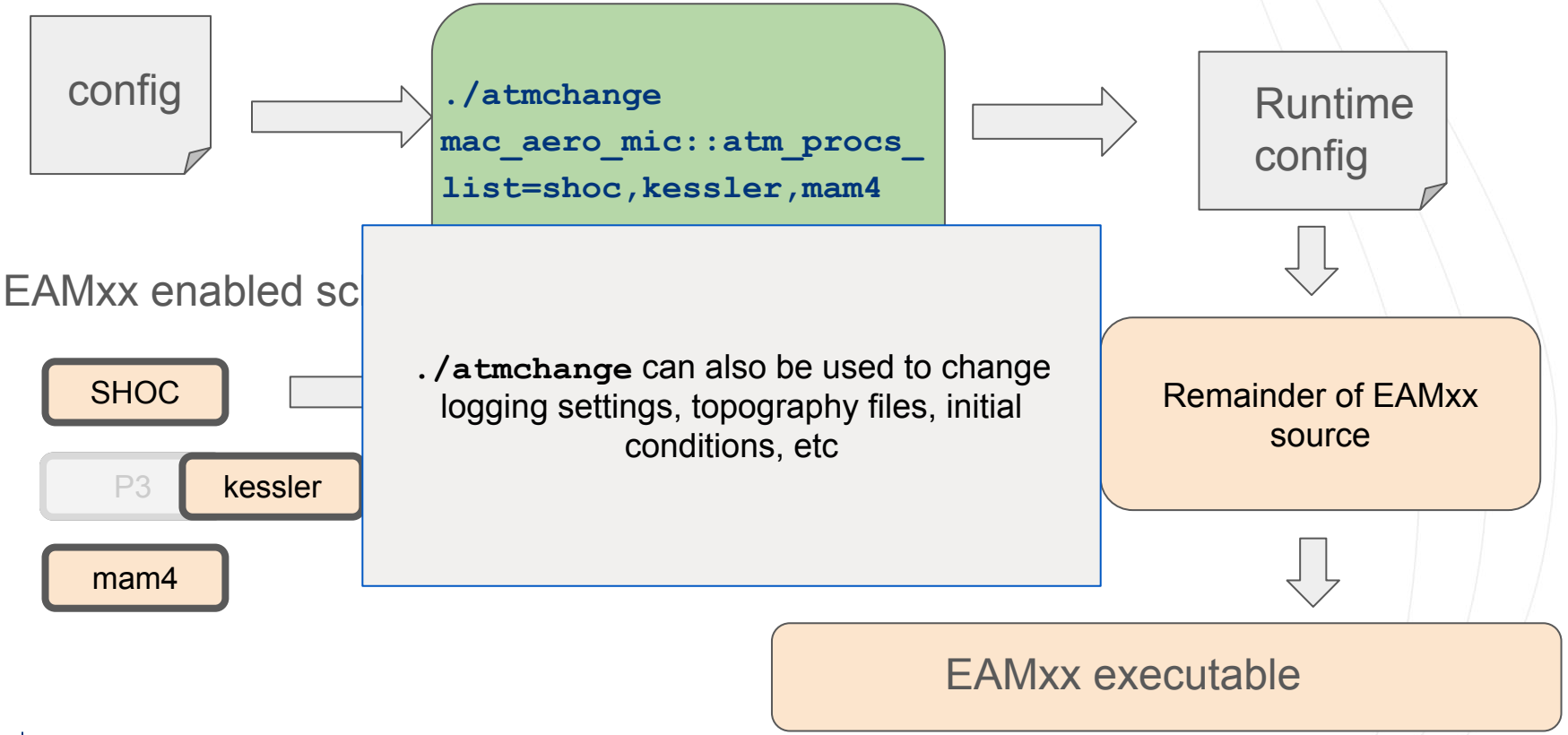
EAMxx change atmospheric processes



EAMxx change atmospheric processes



EAMxx change atmospheric processes



EAMxx automation & sanity checks

- EAMxx handles grid management & time stepping
- Pre- and post-condition checks for all fields
  - Built-in or user defined checks
  - Embeds domain specific knowledge and improves debuggability
- Automatic data handling
  - File IO
  - Data field management between processes

```
Property check 'precip_liq_surf_mass lower bound check: 0' result: Pass
Property check 'precip_ice_surf_mass lower bound check: 0' result: Pass
Property check 'eff_radius_qc within interval [0, 100]' result: Pass
Property check 'eff_radius_qi within interval [0, 5000]' result: Pass
Property check 'eff_radius_qr within interval [0, 5000]' result: Pass
Property check 'NaN check for field T_mid' result: Pass
Property check 'NaN check for field qv' result: Pass
Property check 'NaN check for field qc' result: Pass
```

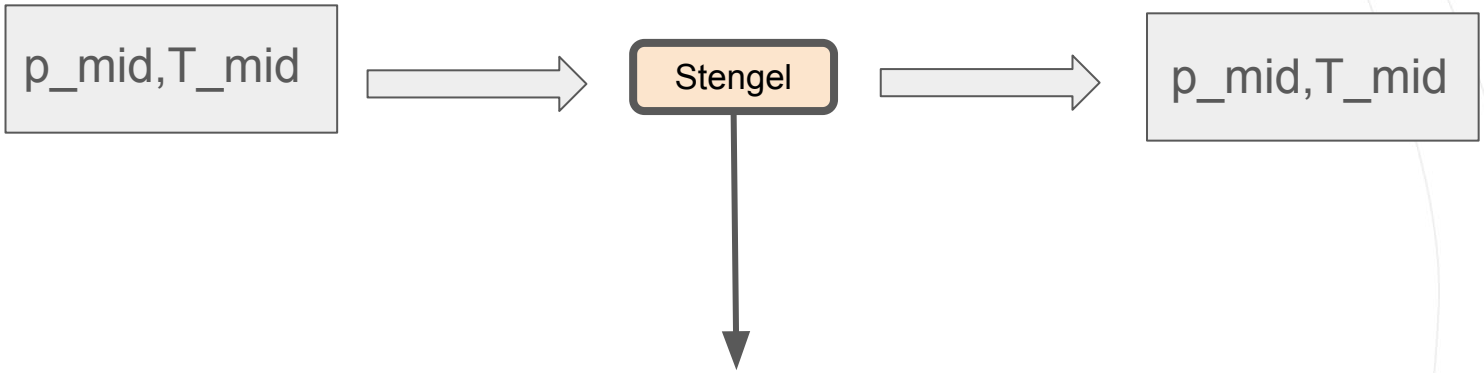
EAMxx parameterization interface overview

- EAMxx parameterizations require the following interface functions:

constructor()	set_grids()	initialize_impl()	run_impl()	finalize_impl()
Usually empty	Define fields and grid layout	Create field checks Initialize things	Run physics!	Usually empty

EAMxx parameterization interface example

- Use an identity transform parameterization as an example

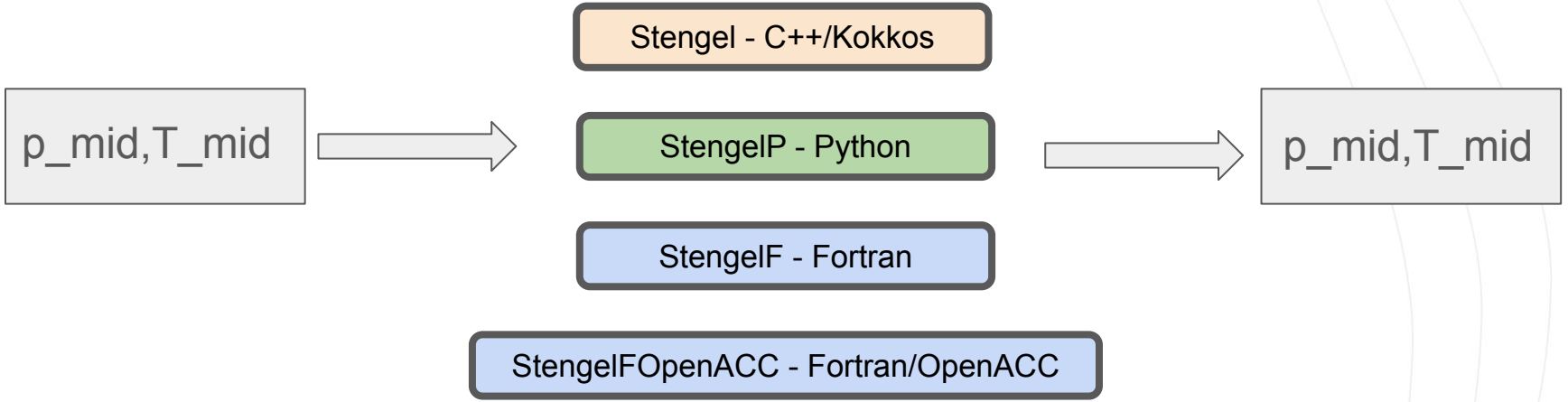


Pressure and temperature at the midpoints (p_mid, T_mid)

p_mid = 2 * p_mid; T_mid = 2 * T_mid
p_mid = 0.5 * p_mid; T_mid = 0.5 * T_mid

EAMxx parameterization interface example

- Use an identity transform parameterization as an example
- Parameterizations can be written with bridges to code in other languages



Pressure and temperature at the midpoints (p_mid, T_mid)

p_mid = 2 * p_mid; T_mid = 2 * T_mid
p_mid = 0.5 * p_mid; T_mid = 0.5 * T_mid

EAMxx parameter interface - C++/Kokkos

set_grids()

m_grid = gm->get_grid("physics");

add_field<Updated>("T_mid", ….);
add_field<Updated>("p_mid", ….);

initialize_impl()

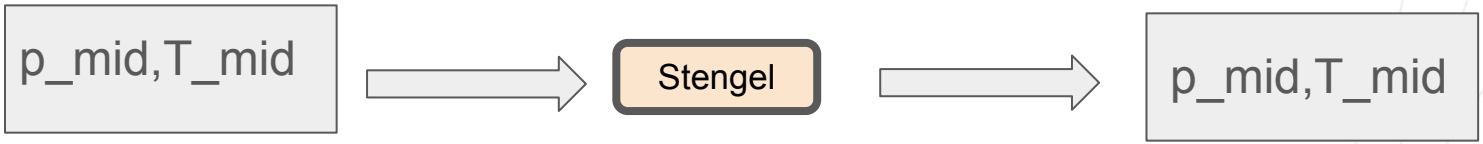
Nothing to do here.

run_impl()

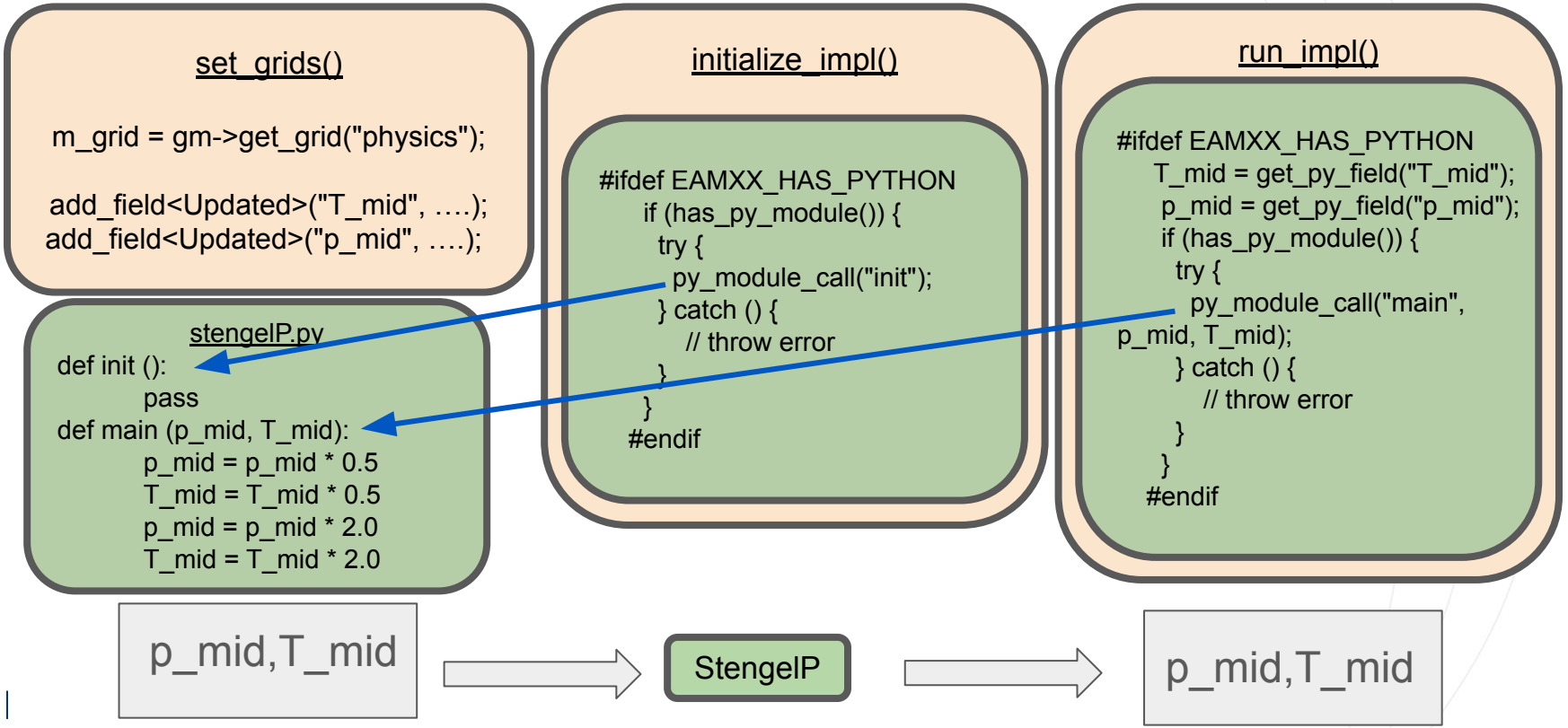
T_mid = get_field_out("T_mid");
p_mid = get_field_out("p_mid");

T_mid.scale(2.0);
p_mid.scale(2.0);

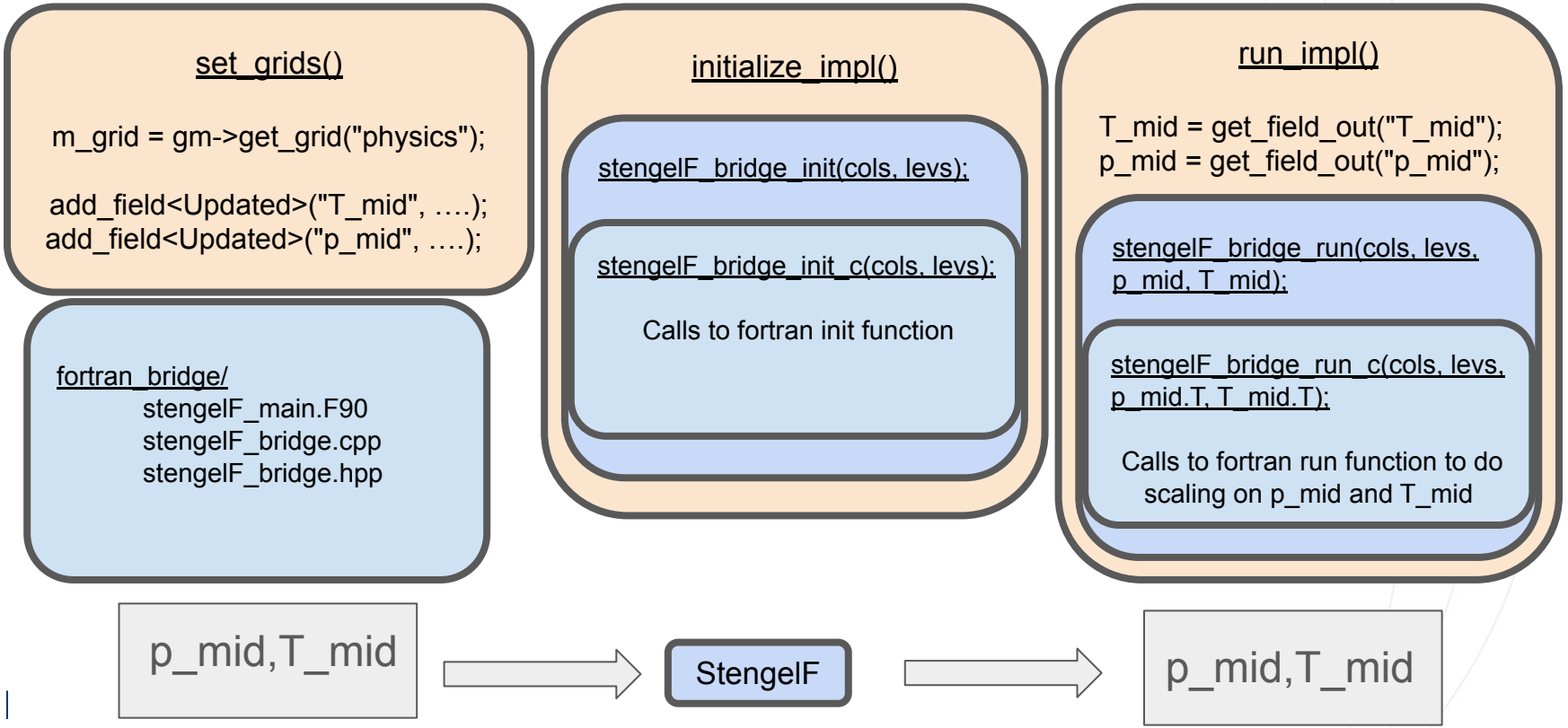
T_mid.scale(0.5);
p_mid.scale(0.5);



EAMxx parameterization interface – Python bridge



EAMxx parameter interface – C++ to Fortran bridge



EAMxx parameterization interface bridges

- C++ and Fortran have different data layouts
  - Requires doing a transpose on fields before going to/from Fortran
  - Use a struct to keep track of C++ and Fortran data fields

- Supports using OpenACC Fortran directives
  - Both the C++ and Fortran code then runs on GPU instead of transferring to CPU for Fortran

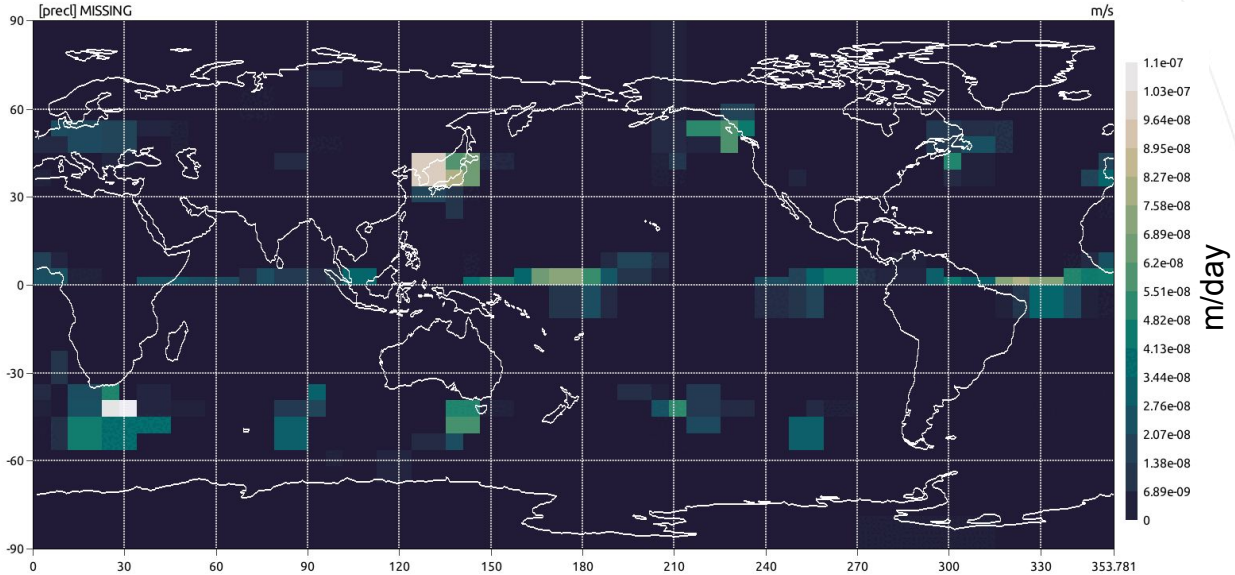
- Can bridge directly to existing code from other repositories! (i.e. CLUBB, PUMAS)

EAMxx with Kessler - WIP

- Kessler from CAM-SIMA
  - Small microphysics parameterization (~783 lines, including meta)
  - CCPP generated code: O(1000s) lines
- EAMxx bridge code written:
  - 1202 lines of C++, including debugging code and comments
  - 179 lines of Fortran for bridge
- Time spent:
  - 2 months for all three Stengel* parameterizations (technical plumbing)
  - Around 2 months for Kessler code
    - I am not a domain expert -> spent a lot of time confused about matching variables, etc
    - Code is currently functional*

Next steps

- Verify Kessler in EAMxx versus FKESSLER in StormSPEED
  - Requires adding analytical conditions capabilities to EAMxx to run the Kessler simulations



Next steps

- Verify Kessler in EAMxx versus FKESSLER in StormSPEED
  - Requires adding analytical conditions capabilities to EAMxx to run the Kessler simulations
- Additional CAM physics parameterizations in EAMxx framework
  - CLUBB
  - PUMAS
- Explore additional language bridge functionality
  - Python/JAX
  - Rust
  - Julia
- Automating bridge code generation

Next steps - portability & performance

- Finish GPU enabling Kessler using OpenACC directives
- Compare memory footprint for EAMxx w/Kessler versus FKESSLER
- Compare memory footprint & transfers and performance for EAMxx Kessler on CPU, CPU/GPU, and GPU

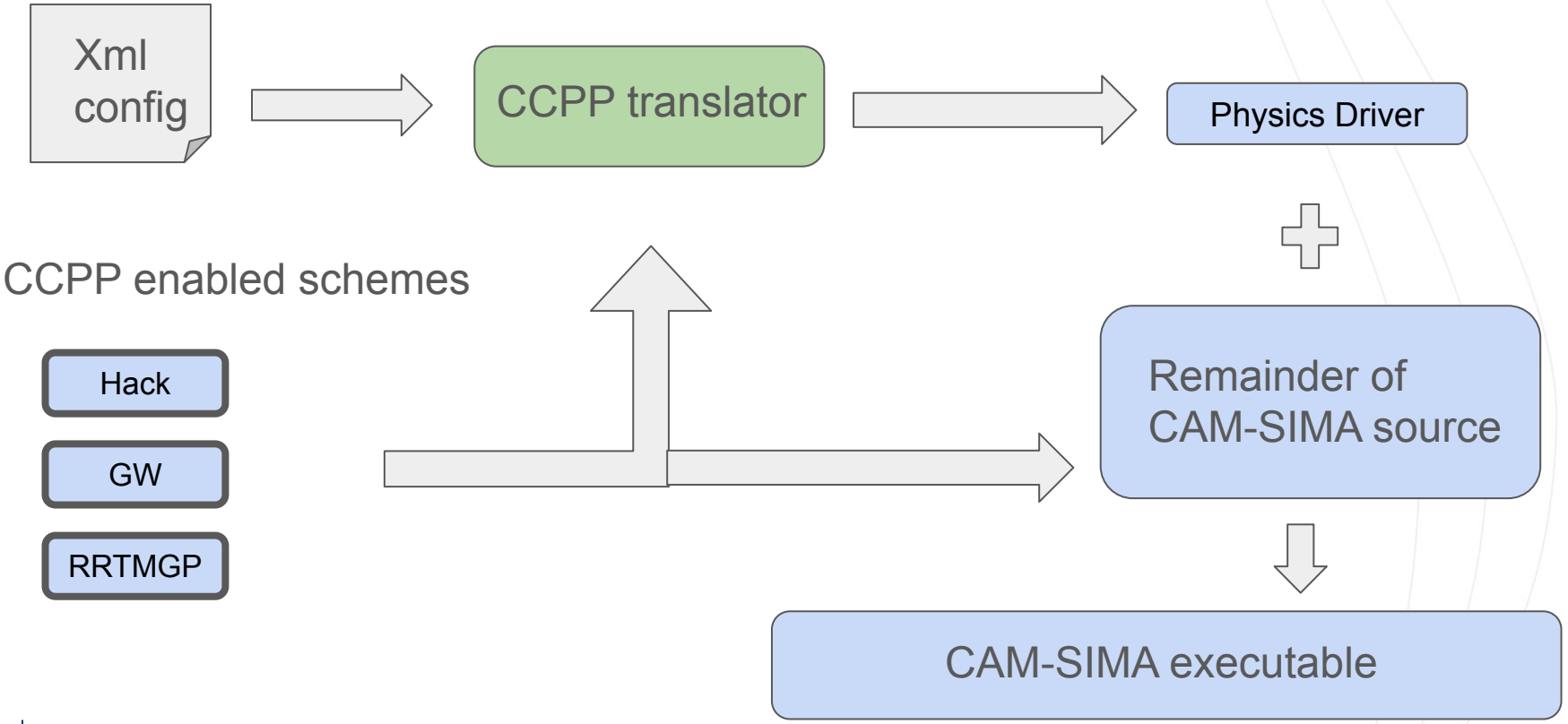
Questions?

Contact: kstengel@ucar.edu

Github branch for the Stengel* example parameters
*Contains Stengel (C++ only), StengelF (Fortran w/OpenACC support), and StengelP (Python) as subdirectories

Supported through the Community Software Facility (CSF)

CAM-SIMA change atmospheric processes



EAMxx parameter interface – C++ to python bridge

set_grids()

```
m_grid = gm->get_grid("physics");

add_field<Updated>("T_mid", ….);
add_field<Updated>("p_mid", ….);
```

initialize_impl()

```
#ifdef EAMXX_HAS_PYTHON
      if (has_py_module()) {
        try {
          py_module_call("init");
        } catch () {
            // throw error
        }
      }
    #endif
```

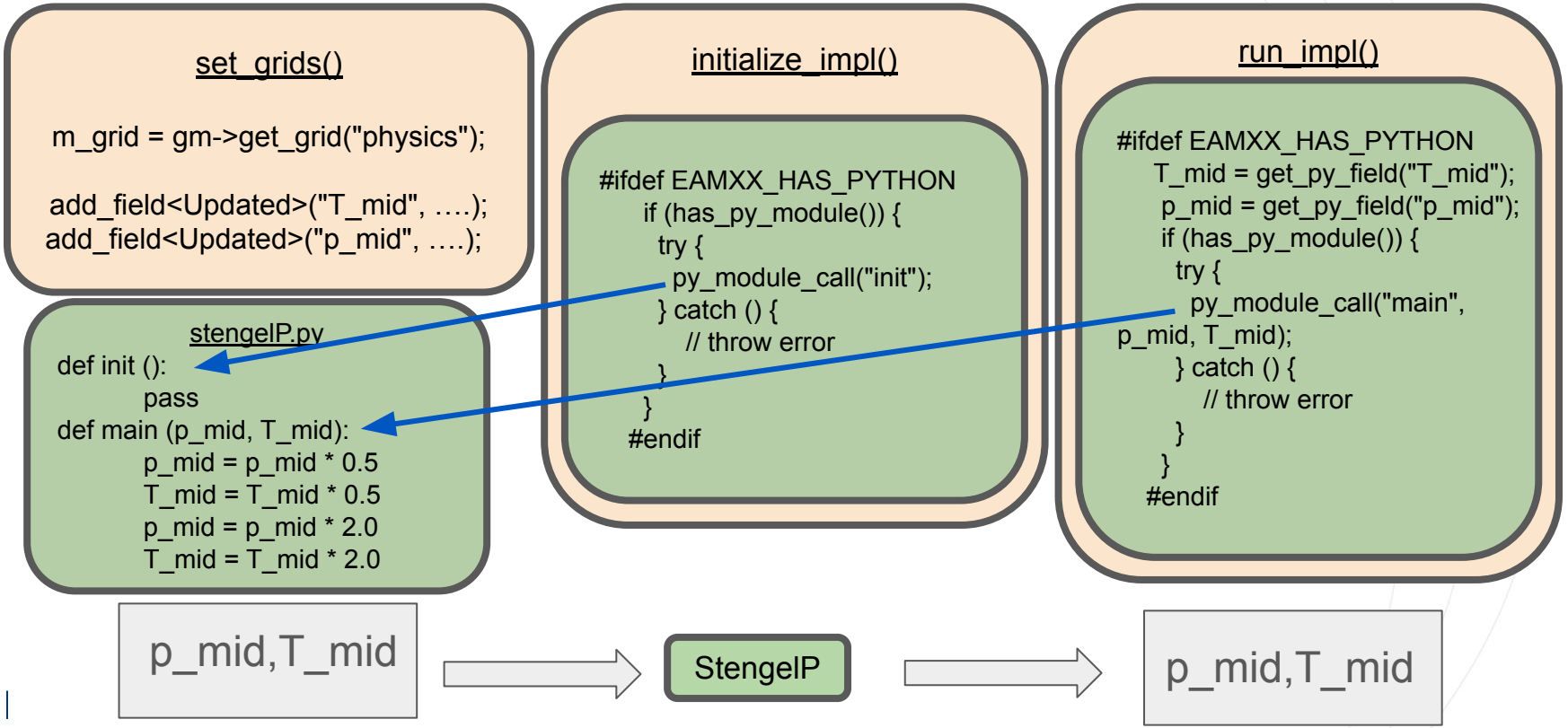
run_impl()

```
#ifdef EAMXX_HAS_PYTHON
      T_mid = get_py_field("T_mid");
       p_mid = get_py_field("p_mid");
       if (has_py_module()) {
         try {
           py_module_call("main",
p_mid, T_mid);
        } catch () {
            // throw error
        }
      }
    #endif
```

stengelP.py

```
def init ():
        pass
def main (p_mid, T_mid):
        p_mid = p_mid * 0.5
        T_mid = T_mid * 0.5
        p_mid = p_mid * 2.0
        T_mid = T_mid * 2.0
```

EAMxx parameter interface - C++ to python bridge



EAMxx with Kessler – GPU results

- Put in CPU and GPU figures if we have them; would want CPU only, CPU/GPU, and all on GPU if possible. Want to show matching results between the three

EAMxx change atmospheric process settings

- Use the **`./atmchange`** functionality to change atmospheric process settings at runtime:
  - Change logging levels:
    - **`./atmchange atm_log_level=debug`**
  - Change overall physics processes:
    - **`./atmchange physics::atm_procs_list=mac_aero_mic,rrtmgp`**
  - Change the parameterizations & their ordering:
    - **`./atmchange mac_aero_mic::atm_procs_list=shoc,cld_fraction,spa,kessler`**
    - **`./atmchange mac_aero_mic::atm_procs_list=shoc,cld_fraction,spa,mam4_aci,p3`**

- Can also be used to set topography file or change initial conditions, etc

Longer page title lorem sit ipsum dolor amet nonummy aliquip cilum ornare fames

Lorem ipsum dolor sit amet consectetur

Venenatis diam ornare fames molestie. Enim at molestie vestibulum nunc dictum quis nulla bibendum. Aenean nulla aliquam nec sed. Diam venenatis vestibulum vitae ut sagittis ultricies molestie. Sed purus egestas elementum in dignissim viverra. Amet at laoreet morbi tellus mattis proin velit at. Mi mi nunc lorem in diam convallis facilisis. Id ridiculus sit et nisl proin erat. Massa facilisi dignissim amet id lectus dis hendrerit.

Erat tellus enim enim ut suspendisse augue ut et. Non vestibulum pellentesque pulvinar sed. Sociis neque enim felis sodales sit maecenas amet. Diam venenatis vestibulum vitae ut sagittis ultricies molestie. Sed neque quam morbi lorem vel vehicula nullam augue. Mattis elementum gravida eget non scelerisque consequat imperdiet sed. Mi erat hac urna facilisis facilisis. Mi mi nunc lorem in diam convallis facilisis. Id ridiculus sit et nisl proin erat. Massa facilisi dignissim amet id lectus dis hendrerit.

Shorter page title lorem sit ipsum dolor nonumy

Lorem ipsum dolor sit amet consectetur

Duis rhoncus facilisis enim sit venenatis diam ornare fames molestie. Enim at molestie vestibulum nunc dictum quis nulla bibendum. Aenean nulla aliquam nec sed. Diam venenatis vestibulum vitae ut sagittis ultricies molestie.

Sed purus egestas elementum in dignissim viverra. Amet at laoreet morbi tellus mattis proin velit at. Mi mi nunc lorem in diam convallis facilisis. Id ridiculus sit et nisl proin erat. Massa facilisi dignissim amet id lectus dis scelerisque hendrerit.

Metus sit posuere vitae nam

Erat tellus enim enim ut suspendisse augue ut et. Non vestibulum pellentesque pulvinar sed. Sociis neque enim felis sodales sit maecenas amet.

Sed neque quam morbi lorem vel vehicula nullam augue. Mattis elementum gravida eget non scelerisque consequat imperdiet sed. Aenean nulla aliquam nec sed. Diam venenatis vestibulum vitae ut sagittis ultricies molestie.

Mi erat hac urna facilisis facilisis. Cras eget morbi blandit arcu in purus arcu consectetur. Egestas tellus aenean augue eget sed purus lacus pretium.

Sed purus egestas elementum in dignissim viverra

Amet at laoreet morbi tellus mattis proin velit at. Mi mi nunc lorem in diam convallis facilisis. Id ridiculus sit et nisl proin erat. Massa facilisi dignissim amet id lectus dis scelerisque hendrerit. Aenean nulla aliquam nec sed. Diam venenatis vestibulum vitae ut sagittis ultricies molestie.

Duis rhoncus facilisis enim sit venenatis diam ornare fames molestie. Enim at molestie vestibulum nunc dictum quis nulla bibendum. Aenean nulla aliquam nec sed.

Shorter page title lorem sit ipsum dolor nonumy

Lorem ipsum	Metus sit posuere	Sed purus	Posuere vitae nam
Duis rhoncus facilisis enim sit venenatis diam ornare	Vel vehicula nullam augue.	Rhoncus facilisis enim sit venenatis diam ornare	Sed neque quam morbi lorem vel vehicula
Sed neque quam morbi lorem vel vehicula nullam	Facilisis enim sit venenatis diam ornare fames	Sed neque quam morbi lorem vel vehicula	Venenatis diam ornare fames molestie.
Duis rhoncus facilisis enim	Sed neque quam morbi lorem vel vehicula nullam	Facilisis enim sit venenatis diam ornare fames	Vel vehicula nullam augue
Sed neque quam morbi	Duis rhoncus facilisis enim sit venenatis diam ornare	Sed neque quam morbi lorem vel vehicula	Duis rhoncus facilisis enim sit venenatis diam

Bolder statement tellus enim enim ut suspendisse augue ut et. Non vestibulum pellentesque pulvinar sed. Sociis neque enim felis sodales sit maecenas amet.