A High-Performance Modal Aerosol Dynamics Library Based on MAM

Jeffrey Johnson[1], Peter Bosler[2], Balwinder Singh[3], Hui Wan[3]

EAGLES Computation Team

ESMD PI Meeting, October 2020

[1]Cohere Consulting, LLC
[2]Sandia National Labs
[3]Pacific Northwest National Laboratory

Goals

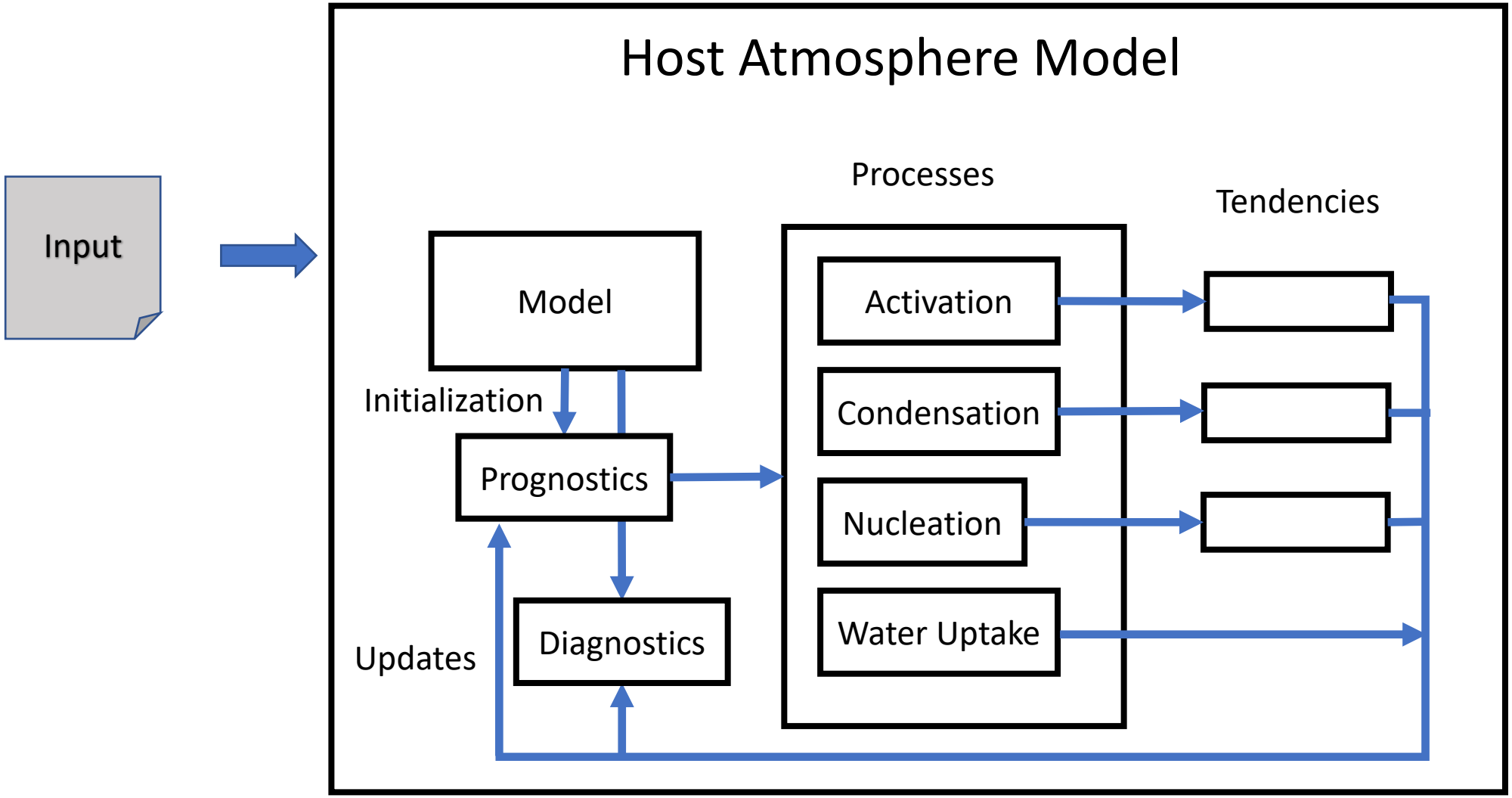
- **Aerosol capability** for E3SM v4
  - “Convection-permitting” parameterizations (~ 1-3km scale)
  - Performance portability (CPUs + threads, GPUs)
  - Verified and validated implementations of aerosol processes
- **Flexibility** to accommodate various aerosol systems
  - Different sets of *modes* (particle size distributions)
  - Different sets of particle *species* that belong to these modes
  - Different *chemical mechanisms* for aerosol-gas + aerosol-cloud interactions
- A **research platform** for improving aerosol parameterizations
  - Support for current and future implementations

Challenges

- Aerosol processes interact with various other processes
  - Different processes for aerosol-radiation interactions and clear-sky/cloud dynamics
  - “Cross-cutting” – not a single sequence of processes
- SCREAM (E3SM’s next-gen atmosphere model) is a moving target
  - We can’t make too many assumptions about how processes are coupled
  - Instead, we *delegate* these decisions to the host model

- What we really need is a set of **building blocks (library)** and a **driver** to verify their correctness and performance.

The Host Model Assembles Building Blocks



The Library Provides the Building Blocks

- **Model** – Stores parameters that define the physical characteristics of an aerosol system and the surrounding atmosphere
- **Prognostics** – Stores prognostic variables that define the system's instantaneous physical state (similar to **`physics_state`**)
- **Diagnostics** – Stores diagnostic variables needed by various parameterizations (similar to **`physics_buffer`**)
- **Tendencies** – Stores time derivatives for prognostic variables at a given time (similar to **`physics_ptend`**). Accumulated into Prognostics during time integration
- **Process** – Implements a parameterization that computes tendencies **or** updates diagnostics for a state at a given time

Processes: Prognostic or Diagnostic?

- A **prognostic process** computes tendencies given prognostic and diagnostic variables:

```
dry_deposition.run(model, t, dt, progs, diags, tends)
```

```
call dust_sediment_tend(ncol, dt, pint, pmid, pdel, t, &
                        dustmr, pvdust, dusttend, sfdust)
```

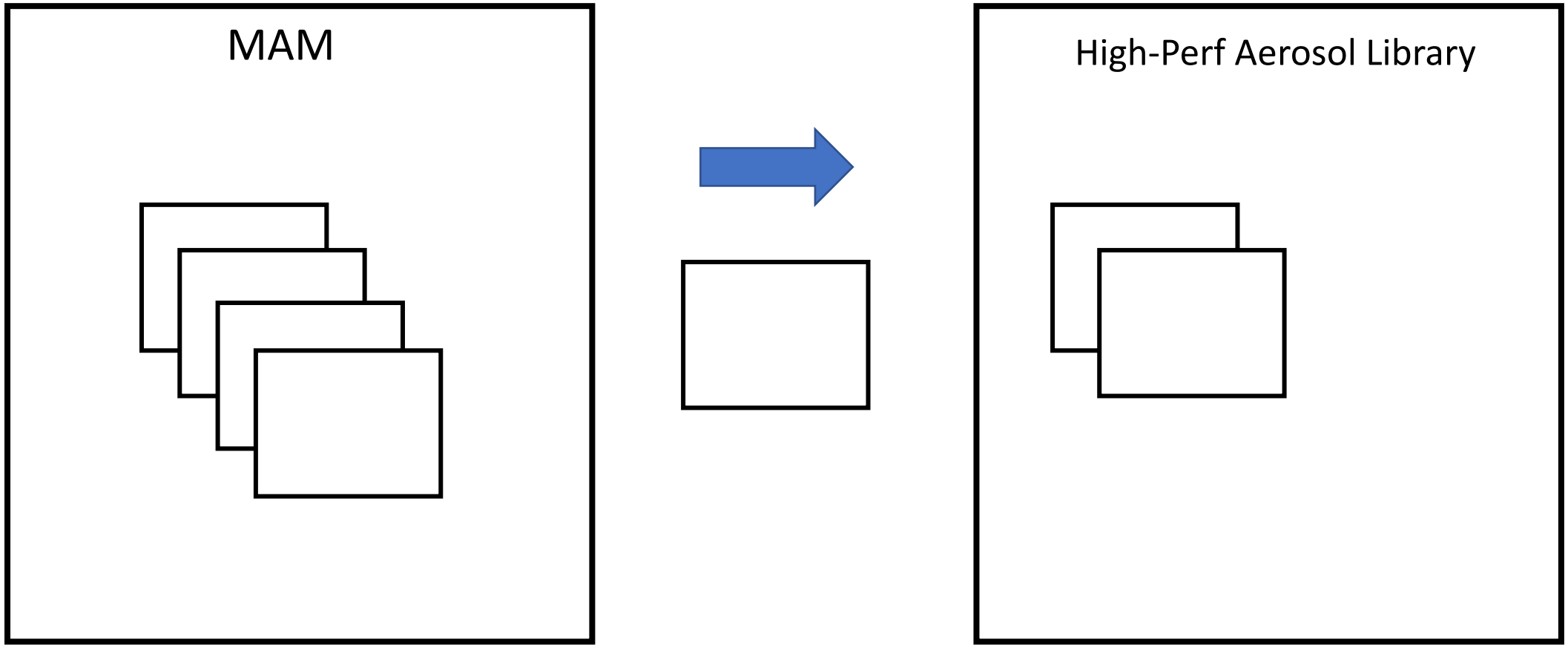
- A **diagnostic process** updates diagnostic variables given prognostic variables:

```
water_uptake.update(model, t, progs, diags)
```

```
call modal_aero_wateruptake_sub(ncol, str_lev, end_lev, nmodes,&
    use_bisection, rhcrystal, rhdeliques, dryrad, naer, hygro, &
    rh, dryvol, drymass, specdens_1, dgncur_a, dgncur_awet,&
    qaerwat, wetdens)
```

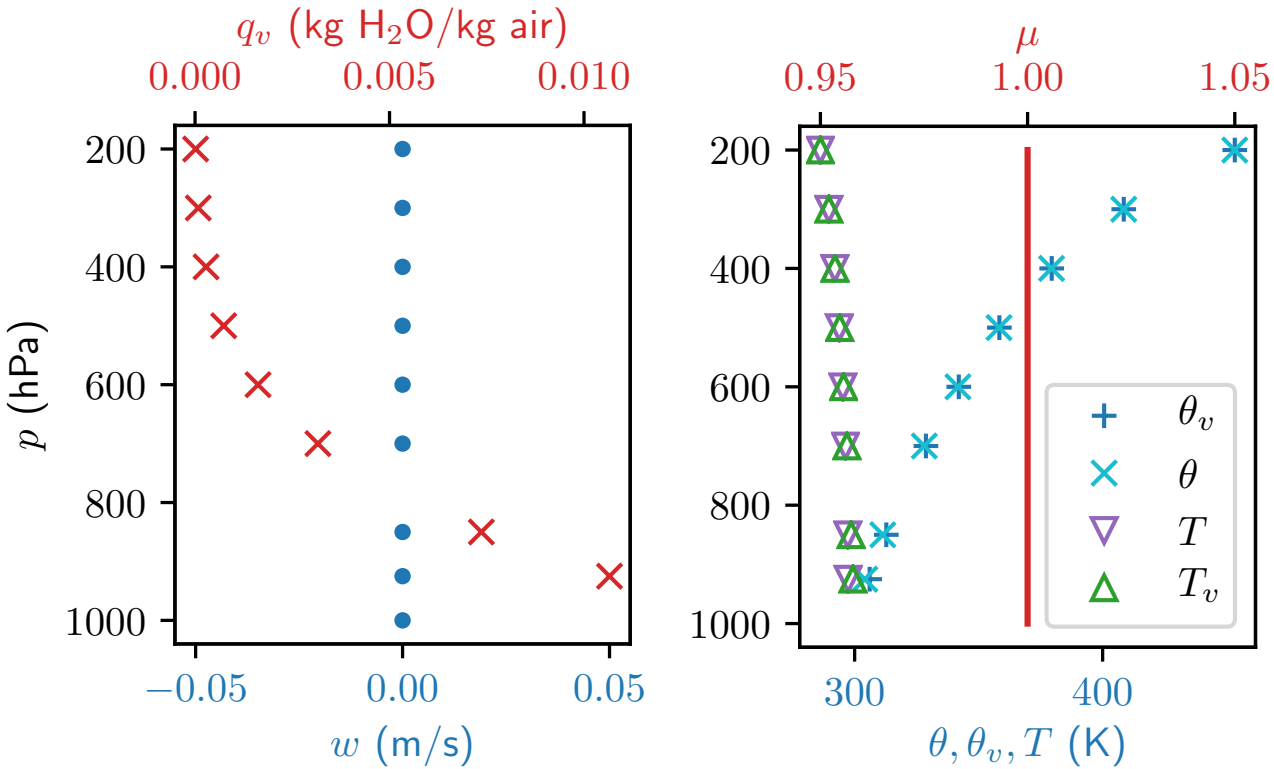
- *A process is a C++ class whose computational work can be done in Fortran or C++/Kokkos.*
- A given aerosol process can have several implementations (C++ classes).
- Variables are extracted from the state and fed directly to functions/subroutines within the process.

Processes Allow Parametrization Transplants!



The Driver Provides a Toy Host Model

- (1D) column model
- Ensembles (multiple *independent* columns)
- Basic 1D dynamics
- Uniform/hydrostatic atmosphere
- Allows selections of parameterizations, perturbed initial conditions
- Useful for experimenting



Next Steps

- Classify all processes in aerosol lifecycle as *prognostic* or *diagnostic*
- Transplant processes from MAM to the new library (in Fortran)
- Devise tests to verify process correctness
- Create C++/Kokkos equivalents to transplanted Fortran processes
- *Verify* C++/Kokkos processes using tests and Fortran implementations
- Implement coupling in SCREAM using the library
- *Validate* processes with global simulations
- Optimize, refine, rinse, repeat!