Climate application requirements for programming models and environments in the Exascale era.

Robert Jacob
Argonne National Laboratory

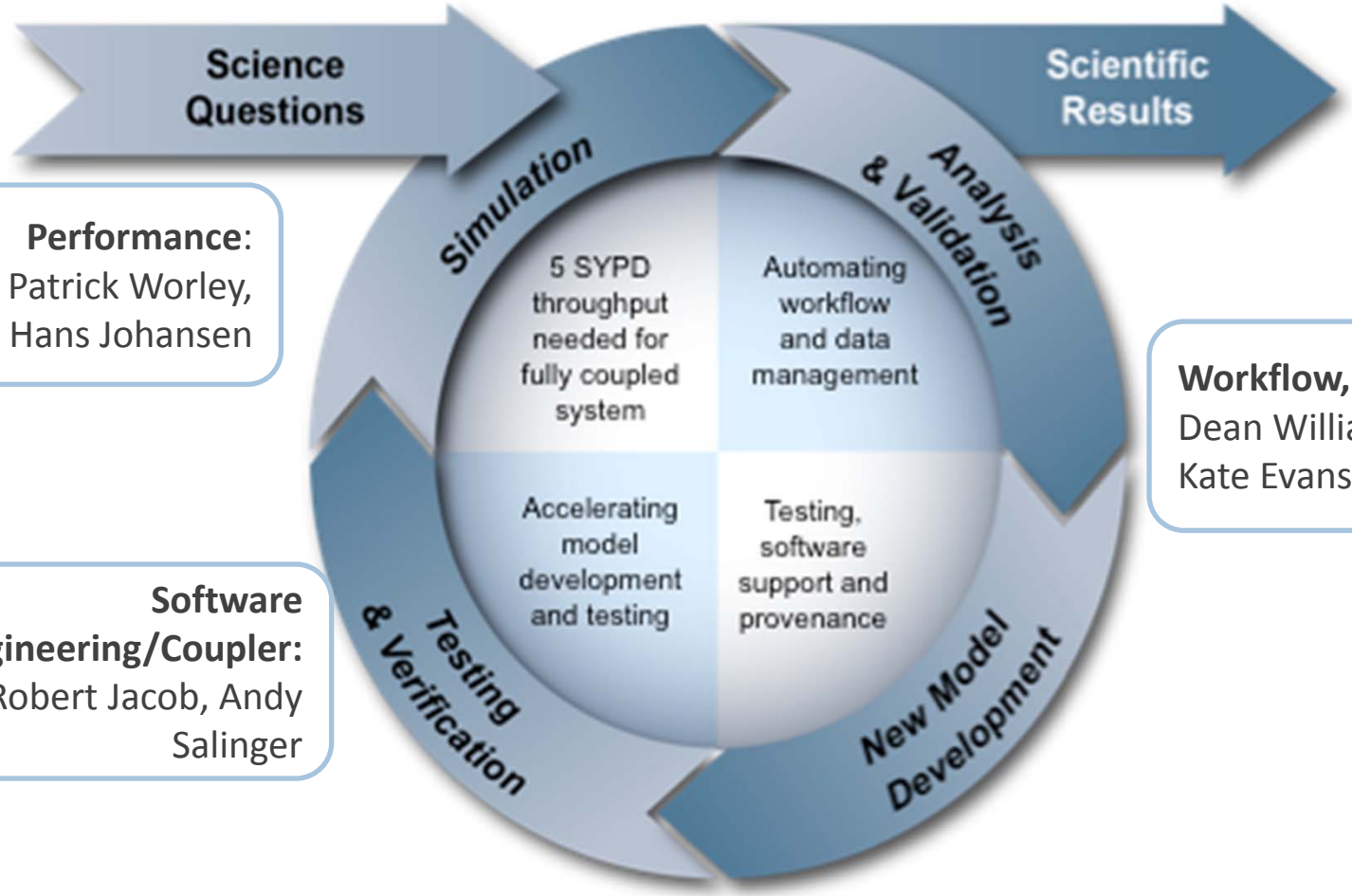
Joint DOE/SC-NNSA ECI PM/E Workshop
March 9-11, Rockville, MD

ACME
Accelerated Climate Modeling for Energy

- **Vision:**
  The Accelerated Climate Modeling for Energy Project is an ongoing, state-of-the-science Earth system modeling, simulation and prediction project that optimizes the use of U.S. Department of Energy (DOE) laboratory resources to meet the science needs of the Nation and the mission needs of DOE
- **Goals:**
  - a series of **prediction and simulation experiments** addressing scientific questions and mission needs;
  - a well documented and tested, continuously advancing, evolving, and improving **system of model codes that comprise the ACME Earth system model**;
  - the ability to use effectively **leading (and "bleeding") edge computational facilities** soon after their deployment at DOE national laboratories; and
  - **an infrastructure** to support code development, hypothesis testing, simulation execution, and analysis of results.

Start date: July 1, 2014
Funding: BER Earth System Modeling program (Dorothy Koch)

ACME's Computational Goal: Accelerate delivery of science results from ACME fully coupled climate model

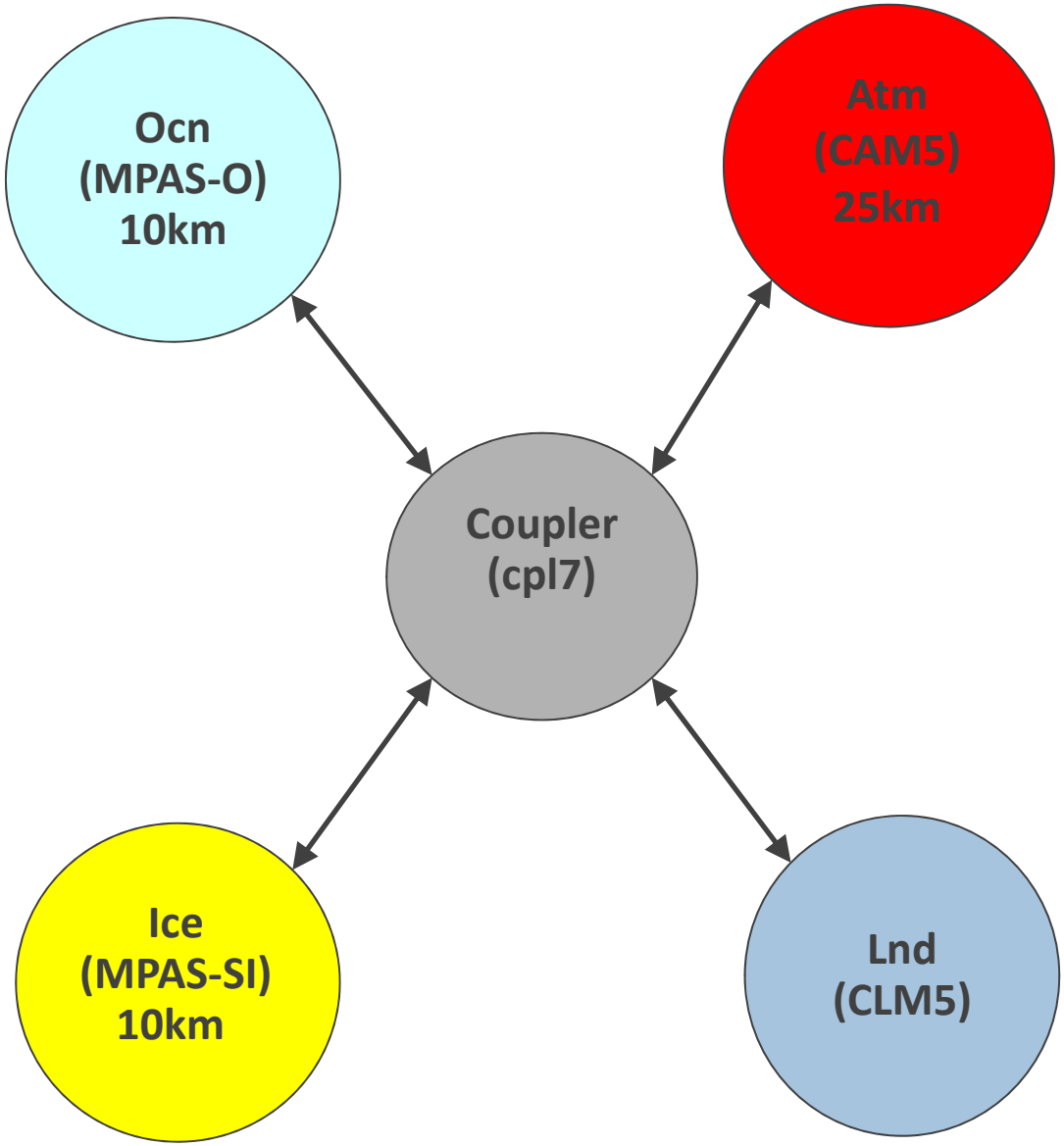


Performance: Patrick Worley, Hans Johansen

Workflow, Diagnostics: Dean Williams, Kate Evans

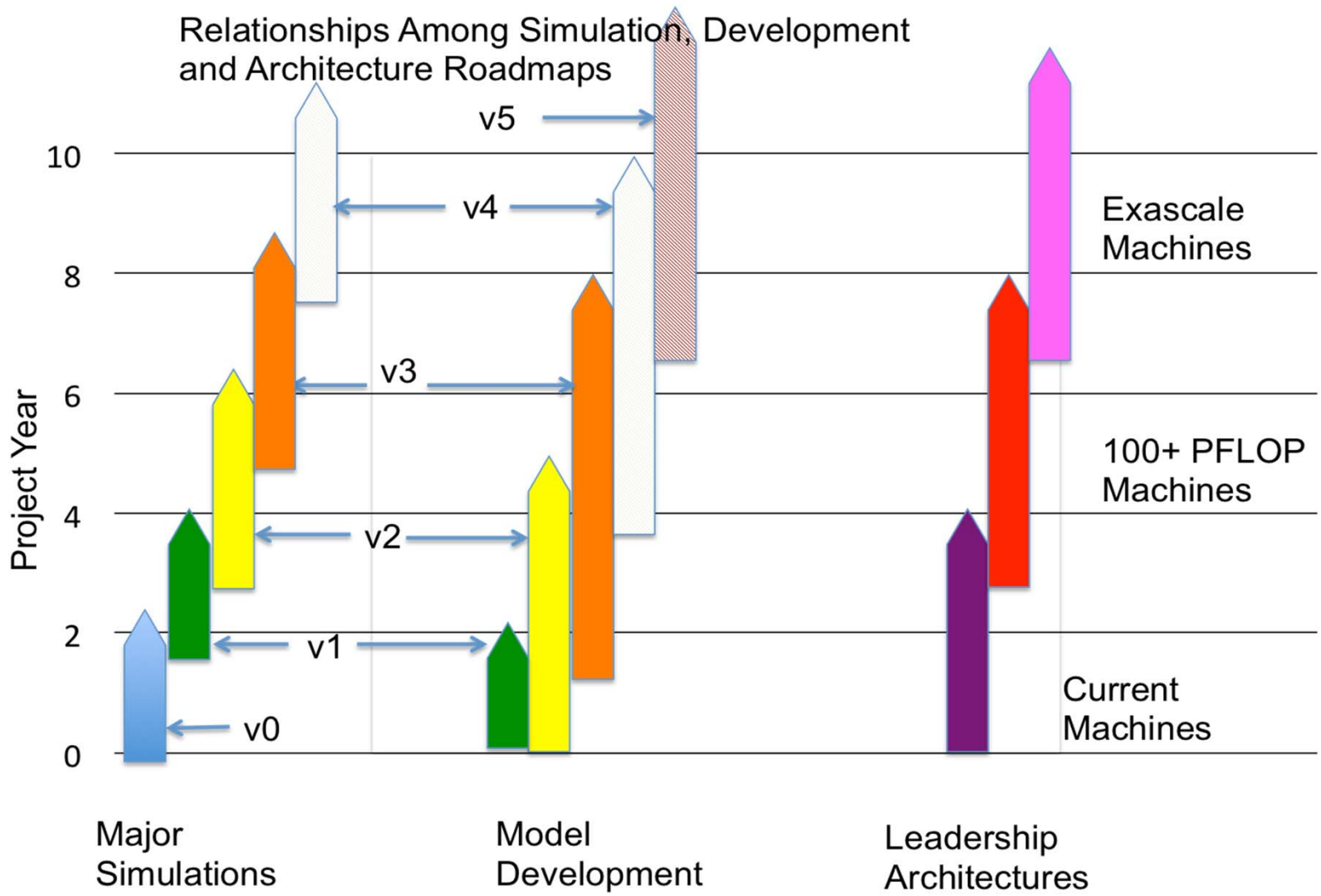
Software Engineering/Coupler: Robert Jacob, Andy Salinger

ACME v1 (planned)



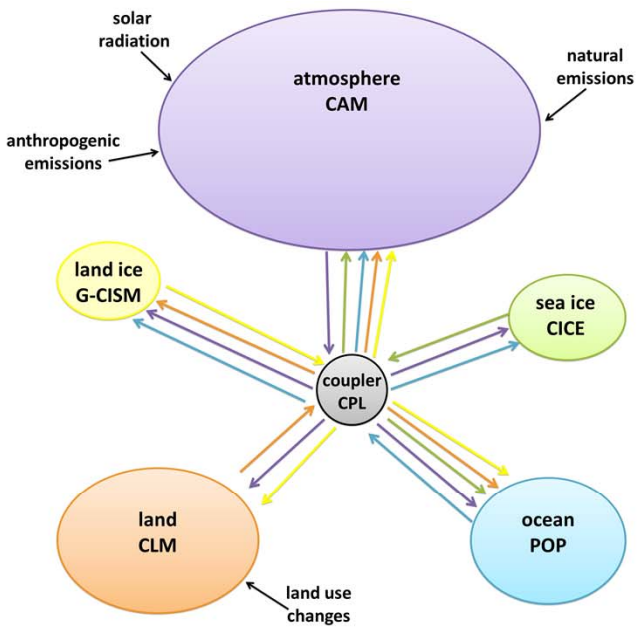
- Approx 1M LOC
  - Mostly Fortran2003
  - Some C, C++
- Multiple (mostly custom) solvers.
- Feature freeze late this year.
- Targeted at current LCFs
- Grid points
  - Horizontal 10^6
  - Vertical 10^2

ACME Roadmap

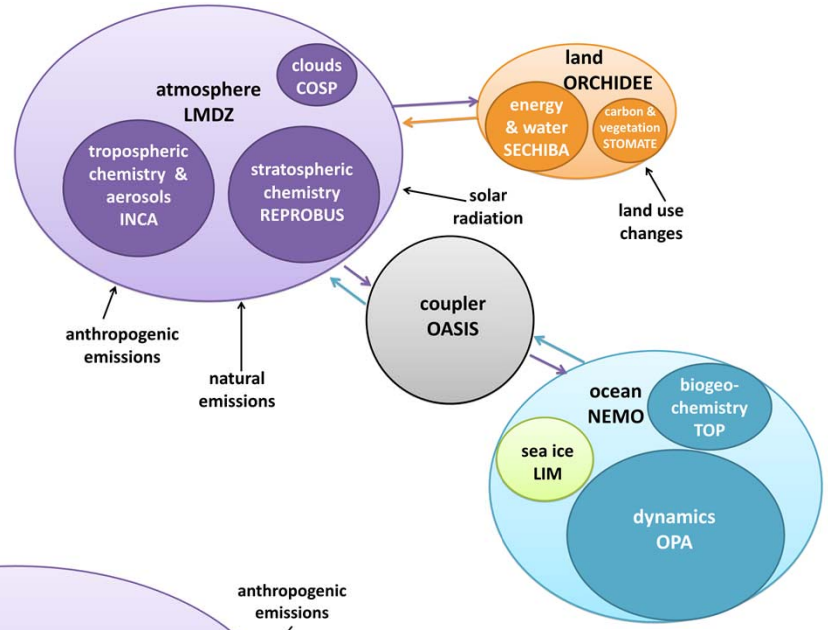


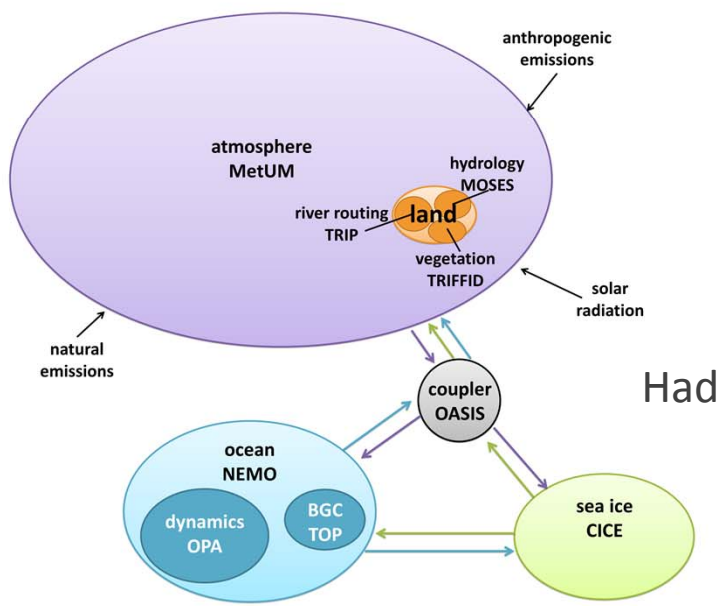
Climate model construction is similar across globe

CESM (USA)



IPSL (France)





HadGEM3 (UK)

Figures from Kaitlin Alexander and Steve Easterbrook. Ovals proportional to code size.

ACME v1 projected performance profile

- Atmosphere continues to be most expensive component with ocean and sea-ice close behind.

- Within all three, tracer-transport module is most expensive. 30-60%.
  - Split equally between: computation, MPI-communication, on-node movement

- Next most expensive is dynamics (similar to tracer transport), then physics

ACME is constructing 2 tracer transport "mini-apps" (one for atmosphere and one for ocean/sea-ice) that can be used for trying new programming models.

- Everything from directives to complete re-write is possible.

- Will take best approaches and implement in tracer transport and dynamics in the full model.

Creating a Coupled Climate Model

- Coupled Climate Models are almost never written from scratch. Use existing Atmosphere, Ocean, land, sea-ice, land-ice models.
- The canonical multi-physics and multi-scale application
- Lots of system-level tests, very few unit (subroutine level) tests.
- Source code for each component (Atmosphere, Ocean) developed separately as a standalone model.
  - Source has different coding styles, conventions.
  - Different internal data structures (for holding computed quantities like temperature.)
  - Wasn't intended to couple with other models.
  - But almost always the same programming language: Fortran.

Note: All following slides are my opinion and not necessarily that of the ACME team.

Climate model exceptionalism (1 of 2)

- While the fluid dynamic equations are settled, the equations for the sub-grid dynamics and non-fluid physics are not at all settled science. New algorithms are created, old ones are tweaked or re-written constantly.
  - Editing the source code to introduce a new algorithm is a regular part of climate modeling. The "solver" is being modified at a low-level.
  - Scientist constantly accessing the data-types and adding instances.
  - (IMO) In other fields, more of the science is in: Formulating the input data, Fitting the domain in memory, Generating the mesh, Controlling numerical error, Analyzing the output, defining/managing the workflow.

- Global atmosphere and ocean are nearly 2-dimensional fluids: 1:1000 aspect ratio. General purpose fluid-dynamics solvers not easily applied.

- V&V wrapped in with code development. Carefully check model output with every line change and compare with previous best solution (bit-for-bit if refactoring). Climate change signal is small so need to understand source of any change in answers.

Climate model exceptionalism (2 of 2)

- Traditionally a flat performance profile: "50 subroutines each taking 2% of the runtime." Depending on exact problem and resolution.
- Many different science subdomains in one model: fluid dynamics AND chemistry AND cloud microphysics AND radiative transfer AND ice sheet flow, etc.
- "Climate model" produces a long multivariate time series of global weather. Climate only understood after extensive data analysis of climate model output.
- Small aspect ratio means 2D plots used for analysis. Little regular use of 3D viz.

Current climate model abstractions used for parallelism (atmosphere-centric):

- "column" - all the points along the vertical direction and a single horizontal grid point. Takes advantage of physical distinction between horizontal and vertical in climate system.
- "chunk" - a group of columns. Could be geographically contiguous or random. Sized for cache.
- "block" – a group of chunks. Could correspond to a contiguous latitude-longitude "block" or be random.
- "tracer" – a substance (CO_2, CH_4) acted on by the velocity field but with no affect on the velocity field itself (passive tracer).
- "Component" – a model of a major physical sub-system of the climate: ocean, atmosphere, land, sea-ice, land-ice.

Currently: MPI + OpenMP machine-level parallel programming model. (for all components)

Static scheduling, two-sided communication

Local view of data and control

components can run on distinct or overlapping groups of processors.

Climate model abstractions NOT currently used for parallelism:

- "level" - a location along the vertical-coordinate. Not used because of fast physical balance in vertical. To much communication required.

- "physics package" - group of algorithms used to calculate non-fluid dynamics forcing and response. Shortwave radiation, longwave radiation. Calculated in sequence but some opportunity for parallelizing-along-physics exits.

New-ish PM abstractions that could be useful (not complete)

- RDMA – for coupling

- Higher level PM's that avoid explicit control of 1B threads.

- Thread scheduling/migration to coincide with known cycles in climate system (daily, seasonal) and how they affect execution.
- Geo-locality aware partitioning strategies across components.

Breakthroughs in PE required for exascale

- Ensemble management and data analysis:
  - Managing runs of dozens-hundreds of members.
  - Analyzing PB of time series data and tracking its analysis.
  - Current practice of using the filesystem as the database probably won't work.
  - Ensembles for UQ offers best path to exascale
- Debugging at scale:
  - Can't run in production at 100K nodes without first debugging at 100K nodes. Same for 1B threads.
- Support from system for asynchronous I/O

Reliance on high-level services

- Need ways to start and manage and communicate between large ensembles
- Need ways to specify component model layout to minimize cross-component communication cost.

Possibilities for task-based and data-parallelism:

- Tremendous possibilities in data post-processing and analysis.
  - "Climate model" produces long multivariate time series or global weather and only post-processing (averaging) yields the climate

Persistent objects:

- Read and write through library calls (NetCDF, PNetCDF, HDF).
- Model AND data analysis tools use same library.

Managing Resiliency

- Already have substantial experience writing user-controlled checkpoints.
- Would prefer OS/PE to handle this either completely or with some interaction with application (with clear signaling), selectable at partition boot-time.
- "Resilient MPI" would be needed if we had to do all this ourselves.

Known unknowns: areas where the climate community needs help.

- Creating new algorithms that solve our science AND expose more parallelism and allow dynamic resource management.
- Learning from other communities (especially other fluid-dynamics and multi-physics communities.)
- Designing code for wide and deep memory hierarches
  - Willing to use directives/compiler hints on our own.

Some climate modeling requirements

- A “reproducible” mode on exascale machines where repeated runs of the same executable on the same number of threads is guaranteed to produce bit-for-bit identical answers. For process-level development, not necessarily production.
- Programming model and environment that allows frequent and deep modification/replacement of highly specific/detailed algorithms.
- “support for teams” e.g. MPI communicators
- “mechanisms [that] permit the automated localized rewriting of software to use DSLs”
- “hardened, tested and supported production-quality software”