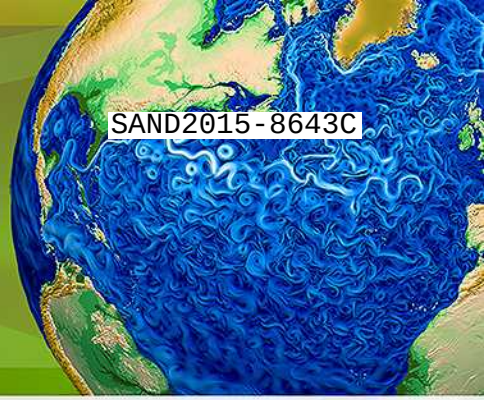
ACME
Accelerated Climate Modeling for Energy



The ACME project's dycore performance strategy for next generation architectures

Mark Taylor

Sandia National Laboratories

mataylo@sandia.gov

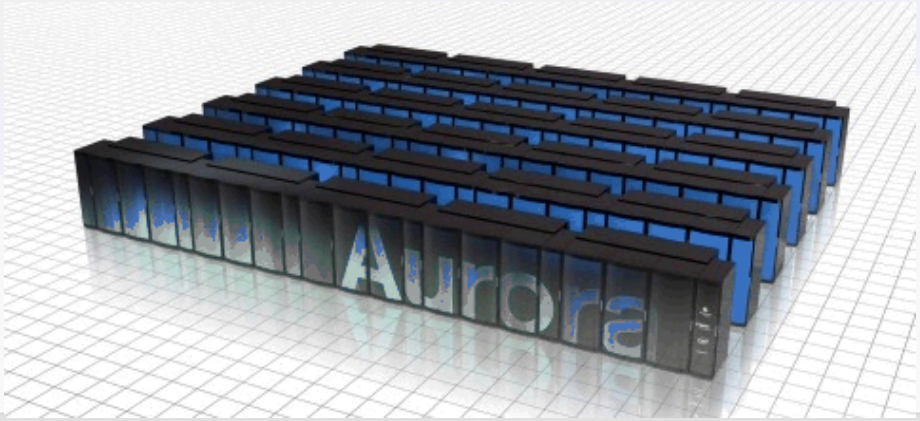
U.S. DEPARTMENT OF ENERGY

Abstract

- The ACME project is building an Earth System Model designed to run efficiently on DOE's next generation architectures. We are currently targeting two types of upcoming near-exascale platforms: An IBM Power9/NVIDIA GPU system and Intel Phi/KNL systems. In the near term, we are relying on openACC for GPU systems and openMP for KNL systems. Both of these approaches need extensive data layout and loop restructuring in order to obtain reasonable performance. It is expected for key computationally intensive kernels, we will maintain two versions, with the openMP version suitable for KNL and generic CPU systems, and the openACC version for GPU systems. Longer term, we hope to adopt a performance portable approach which will allow us to obtain reasonable performance on both types of systems in a single code base. We are using ocean and atmosphere transport mini-apps to evaluate two such approaches, the Kokkos C++ array classes which allow us to abstract both the data layout and the execution model, and Legion, a programming model designed around task based programming model.

ACME Next Generation Machines

- Key goal of ACME: Run on next generation DOE machines:
- NERSC Cori (late 2016)
  - 9300 Nodes
  - Intel Phi KNL, >60 cores each
- OLCF Summit 2018
  - 3400 Nodes
  - Multiple IBM power9 and NVIDIA GPUs
- ALCF Aurora 2018
  - 50K Nodes, 3rd gen Intel Phi



ACME on today's systems

- NERSC Edison
  - 5600 Nodes,
  - 24 cores (dual Intel Ivey Bridge)
  - Watercycle Prototype: 3.1 SYPD
- OLCF Titan
  - 19K Nodes.
  - 16 core AMD Opteron + NVIDIA GPU
  - Watercycle Prototype: 2.2 SYPD
  - Difficult to get sufficient INCITE allocation because we don't yet fully utilize the GPU
- ALCF Mira
  - 49K Nodes
  - 16 core PowerPC
  - Watercycle Prototype: 0.9 SYPD
  - Need better scaling, or can run effectively via ensembles
- All ACME v1 simulations will have to be done on these machines!

Exascale Programming Model

- Programming model for 100PF systems (Aurora/Cori/Summit):
  - MPI + openACC/openMP
  - openACC/openMP may merge -> openMP4
- Programming model for Exascale?
  - Unlikely to change for Aurora/Cori/Summit follow-on machines
  - We actually *do* know the programming model for Exascale. P. Messina – recent presentations to ACME and A2E

Exascale Programming Model: Caveats

- Reasonable performance will require specific data structures and loop structures
- If optimal structures are different between GPU, Phi and CPU systems:
  - Maintain multiple versions of the code?
  - Higher level languages/approaches:
    - E.g. Kokkos which can abstract both the data model and the on-node execution model
    - E.g. Legion: Task based programming models which replaces MPI code and/or outer loops as tasks. Tasks can then be distributed among nodes & cores.

ACME Mini-App Strategy

- We need to test these approaches for use in ACME:
  - openMP, openACC, newer programming models
- Getting help from: NESAP/Cori, CAAR/Summit and possibly ESP/Aurora
- Recommendation of ACME/Exascale study group: *Identify key kernels/modules that are small enough so a single person can understand/refactor/rewrite to test new approaches, but that are large enough that successful results are meaningful for ACME.*
- Result: Transport mini-apps for both atmosphere and ocean

ACME Mini-App Strategy
Why Tracer Transport?

- Transport is a relatively small code base (significantly smaller than the dynamics: simpler equations, less terms)
- Yet transport is quite similar to the dynamics:
  - Spherical geometry
  - Nearest neighbor communication
  - Similar differential operators
- Atmosphere and Ocean mini-apps, to cover finite element and finite volume approaches used in ACME
- Atmosphere tracer transport is single most expensive ACME component. Ocean tracer transport is 30% of the ocean model

ACME Mini-App Strategy

- Identify best new architecture strategy in the transport mini-apps
  - Implement in ACME transport
  - Extend to atmosphere/ocean dynamics
  - Extend to MPAS-I
- Represent 70% or more of ACME

ACME Mini-App Strategy
What about (Atmospheric) Physics?

- Huge code base
  - But only 20% of the cost of the atmosphere
  - significant effort to rewrite – only do this if we are *sure*
- Can scale our way out of expensive physics
  - Physics is embarrassingly parallel (at least for now)
  - Example on Mira:
    - we run well past the scaling limit of the dynamics
    - Dynamics only uses 24 threads per node
    - Physics uses 64 threads per node, with each thread having 4-4 physics columns to work on

Summary

- ACME v1 needs to run on existing machines
  - All initial efforts should be those that benefit both existing and next-gen (better GPU usage, better threading, communication hiding, algorithmic speedup)
- ACME is using a “mini-app” strategy to evaluate new approaches