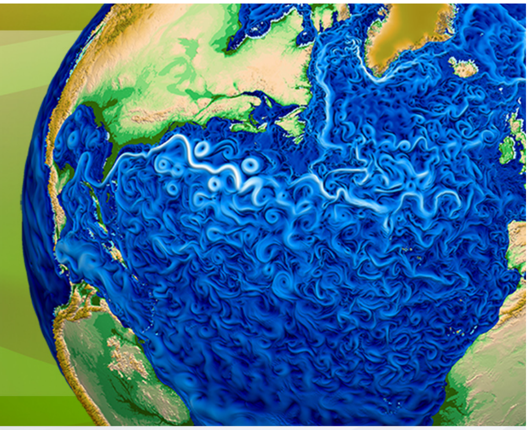
E3SM
Energy Exascale
Earth System Model



Choosing your Candidate in the Programming Model Primary

Phil Jones, LANL
E3SM Performance Group Lead and
PI: Coupling Approaches for Next Generation Architectures (CANGA)

LA-UR-18-28763

U.S. DEPARTMENT OF ENERGY

The exascale wall

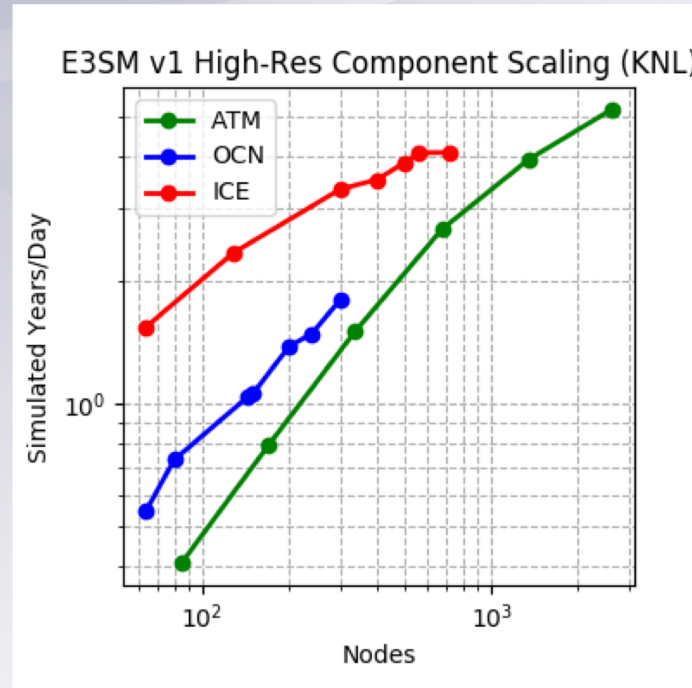
- Vendors building a big beautiful wall
  - Prototypes: Summit, A21, Cori, others
- Searching for primary candidates
  - Provide Programming Model reform to allow legal scientific codes to enter
  - Maintain HPC diversity values:

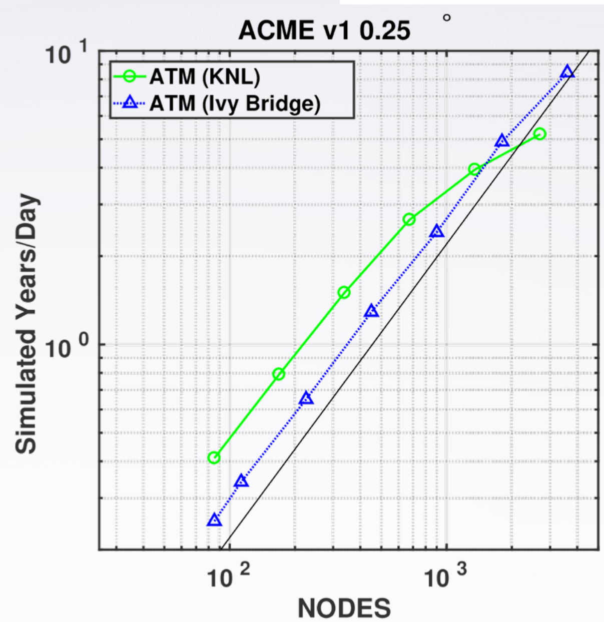
    …Give me your tired, your poor,
    Your huddled developers yearning to reach peak.
    The wretched refuse of legacy dycores…

The Incumbent

- State of E3SM
  - MPI + OpenMP + some OpenACC/GPU
  - Doing a good enough job on KNL
  - Some GPU speedups in a few kernels
  - Strong scaling scandal: Not enough work for more powerful nodes at strong scaling limit
  - More worried about re-election and riding out Moore's law until retirement

Performance and scaling of E3SM components on Cori-KNL

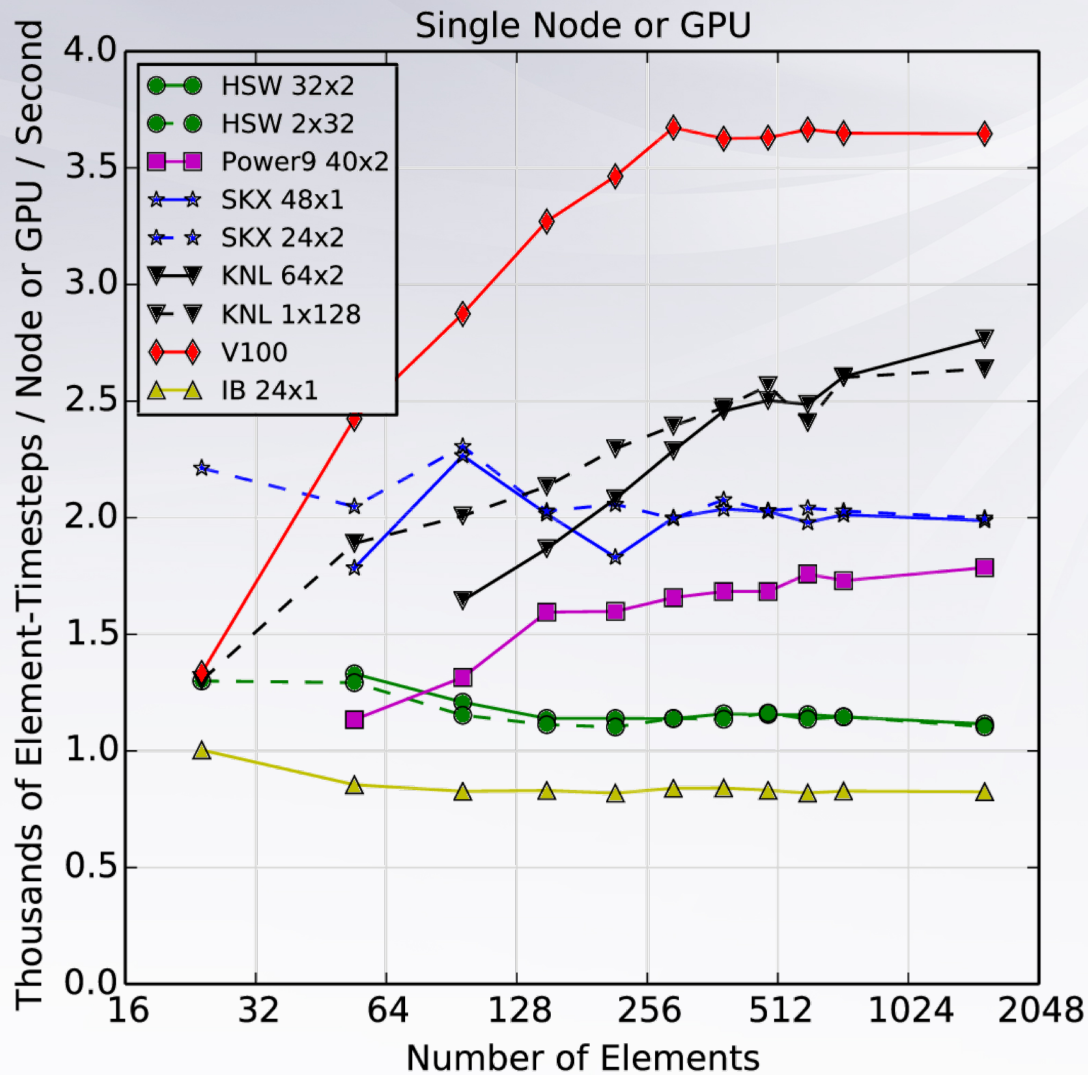




Performance per *node* for high-res atmosphere on Edison (2x12 cores), Cori (68-core KNL)

Atmosphere Dycore - Node Performance

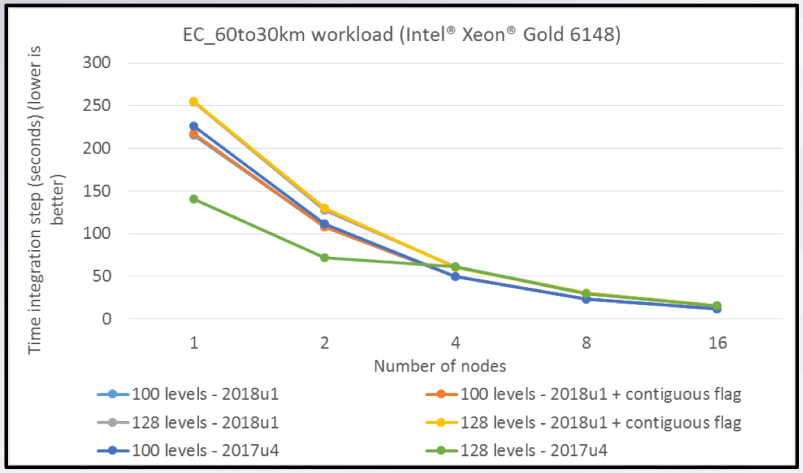
- Kokkos version of dycore illustrates several trends:
- Moore's Law near death
- New architectures (GPU and KNL) only provide speedup in the high work/node regime.
- At high throughput (>1 SYPD): GPU/KNL not helpful.
- Medium throughput (1 SYPD): moderate benefit
- Low throughput (<<1 SYPD): can provide significant benefit but GPU requires significant code investment



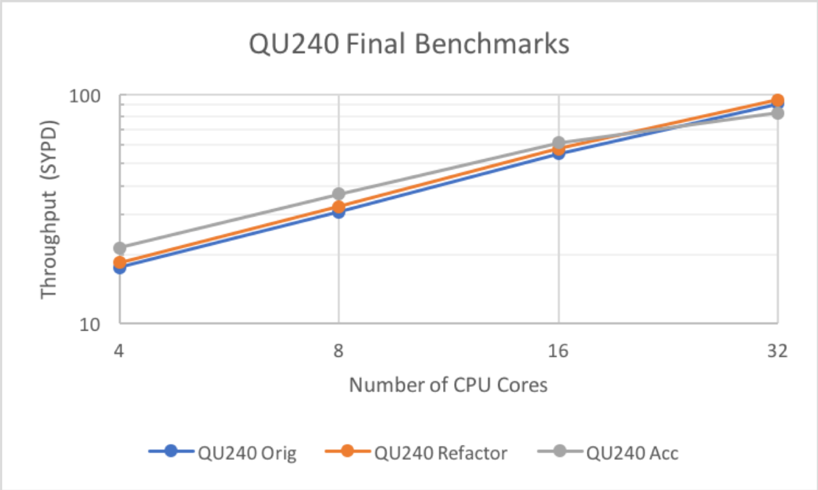
Kokkos dycore across different node types (courtesy M. Taylor, A. Bradley, SNL)

Ocean (MPAS-O) similar

- Vector, threading and other optimizations
  - Good improvement at low node counts with high workload
  - Strong scaling limit shows little improvement
- GPU optimization
  - Speedups of 2-4x for some kernels on GPU
  - Speedups disappear at typical production scaling
  - Shows lack of kernel



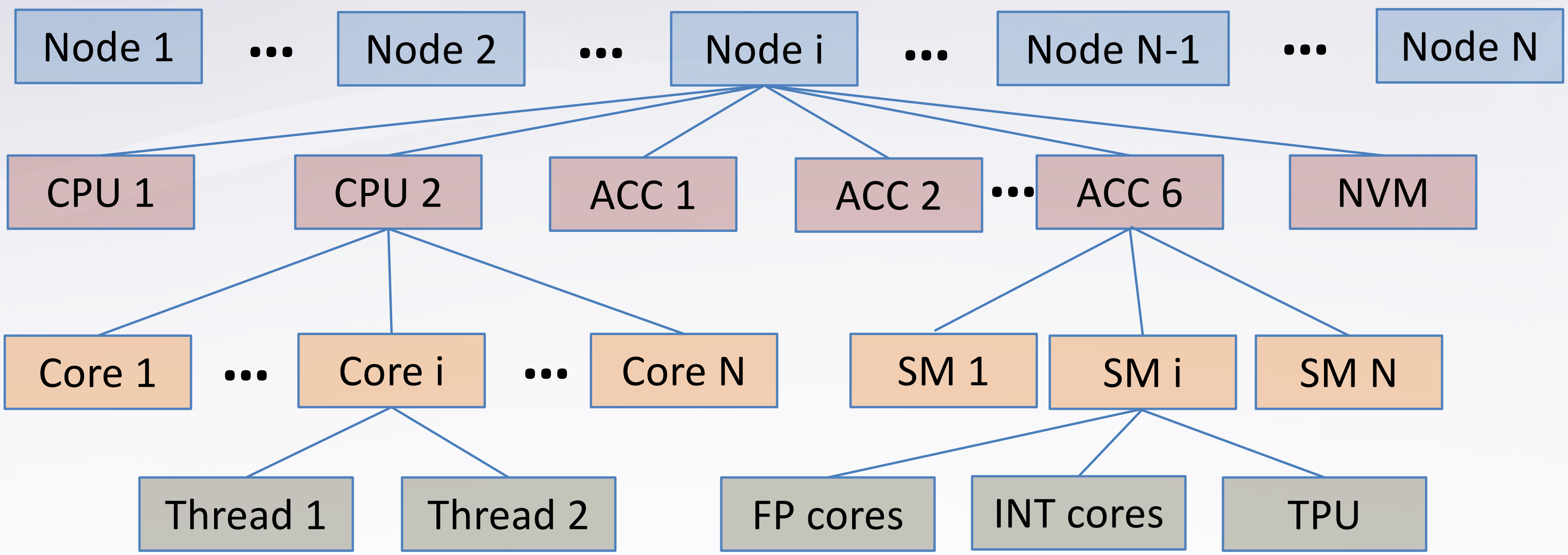
KNL improvements by N. Hariharan, J. Do, Intel



Performance on summit-dev after optimizing EOS kernel

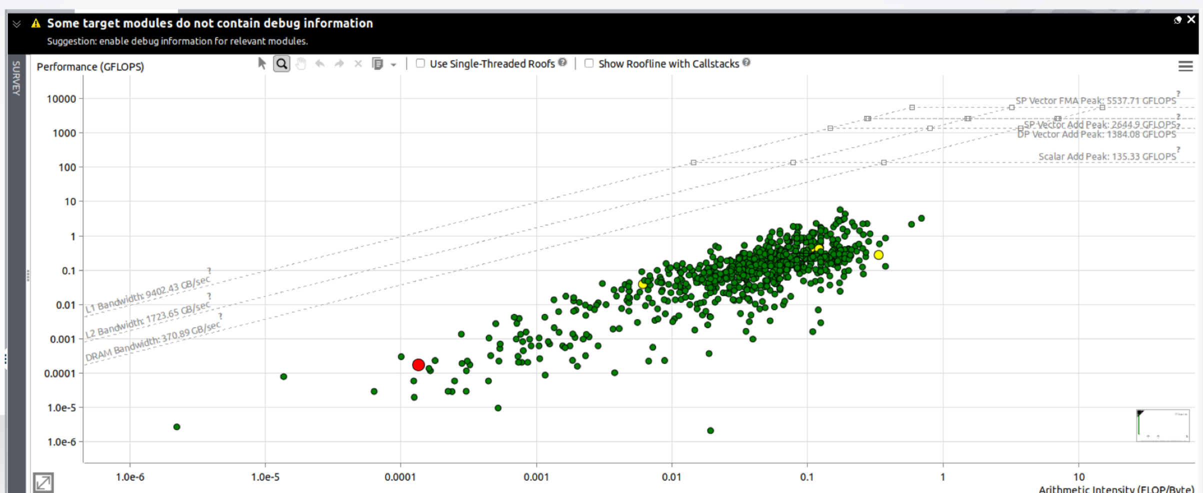
The Party Platforms

- Hardware platforms
  - Parallel at all levels
  - Hybrid at all levels
- Programming models must:
  - Be parallel at all levels
  - Task-parallel at all levels
- Codes must:
  - Supply more work per node



Aside: Purpose Built

- At least two interagency efforts
  - To be public soon
    - pretty far along on exploring new memory features
  - ESPC
    - In spin-up/exploratory stage
- Chip manufacturing
  - Ability/desire(?) to mix/match tiles to differentiate
  - Much more flexibility that doesn’t require $B fab
  - Still focused on larger commercial apps
- Cost effective?
  - Not clear yet

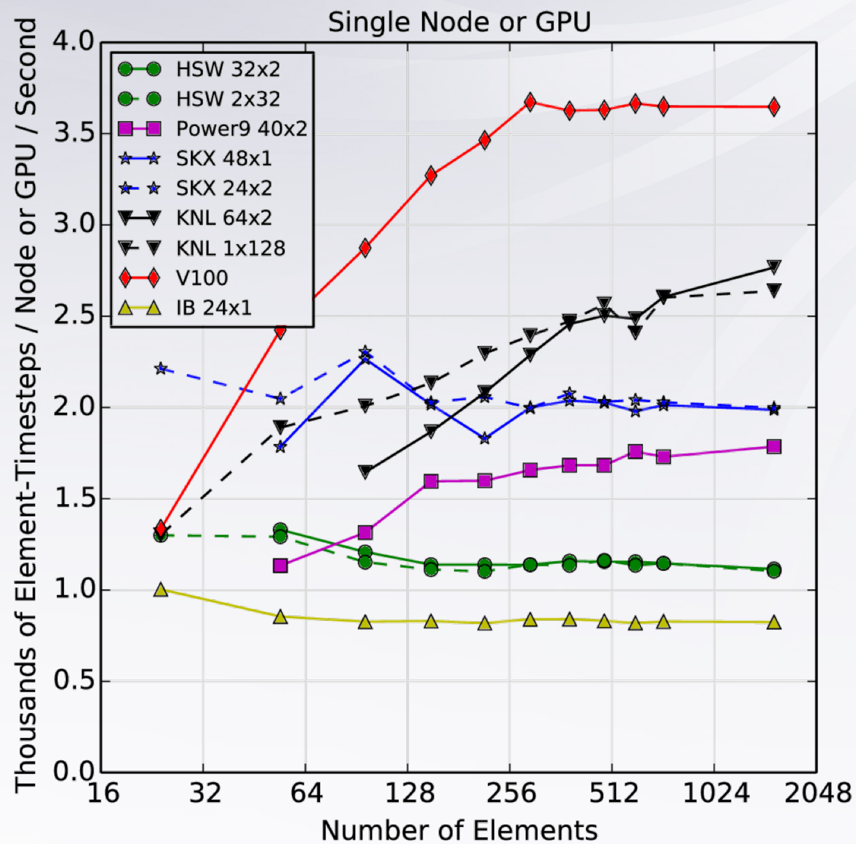


The Establishment Candidate

- MPI+OpenMP/OpenACC
- Good name recognition/ ease of entry
  - Policy ideas relatively well-understood (extensions to OpenMP, etc.)
  - Relatively easy to implement
- Broad constituency
  - Can have different forms, approaches for each kernel
- Some baggage
  - Affinity, memory management
  - Fragility, esp. if threads at high level
  - More responsive to the party, less control for the developer/constituent
  - Not great at multi-tasking (asynchronous execution)

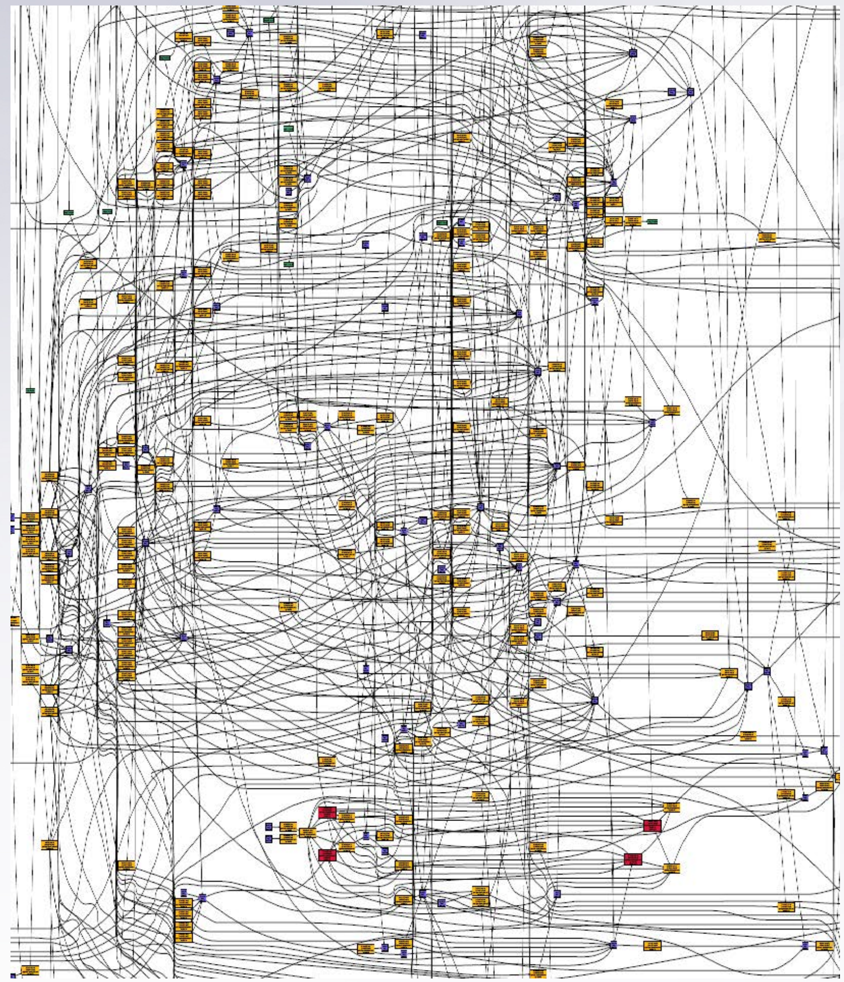
Attacks from left - Kokkos

- Deep state, paternalistic
  - We're a framework and we're here to help
- Separates index space, data model
  - Templated data ordering
- Optimizations for loop-level patterns
- Support for hybrids in host/device
  - Async: parallel for returns before functor end
- Focus on portability aspect of performance portability
- Some success
  - Can achieve performance as good/better than Fortran across CPU architectures
- Weaknesses
  - Requires new taxes, both for buy-in and effort to get performance
  - Become dependent of the state
  - Give up freedoms
  - Probably not enough task parallel support



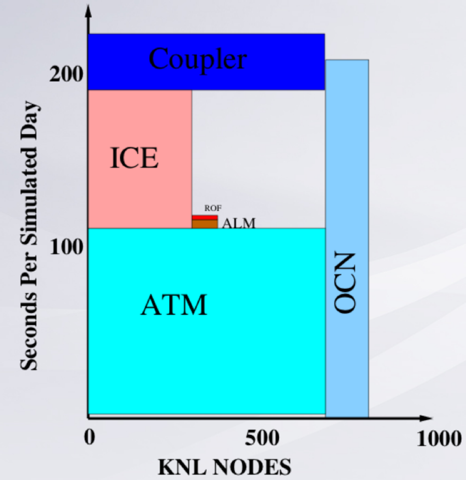
Attacks from right – task parallel

- Formulate model as loose federation of tasks with clear data dependencies
  – Push power to states
- Lightweight, but authoritative govt (run-time)
  – Task queue
  – Dependency graph (DAG)
  – Assigns tasks to exploit all resources available
- Examples
  – **Legion,** Uintah, PARSEC, HPX, others



Example DAG from S3D combustion

Asynchronous Many-Task



- Computing advantages
  - Exposes more parallelism
  - Can automatically load balance
  - Fault tolerance
  - Map tasks to appropriate hardware (I/O, GPU, CPU, etc.)
  - Requires clean interfaces that also enable better testing
- Science advantages
  - Managing complexity: to add functionality, add task to task queue
  - Treat models as collection of processes, couple at process level
  - Move away from large monolithic stove-piped components
  - Scale-aware: launch and couple tasks at appropriate time, space scales, enable more flexible time integration
  - Include more of overall workflow (e.g. in situ analysis)

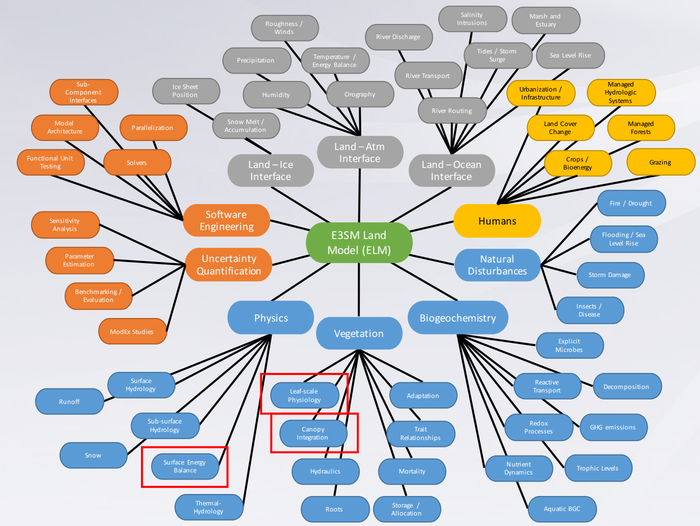
CANGA: Coupling Approaches for Next Generation Architectures

- PIGLET: Prototype Integration using Legion Execution of Tasks
  - Legion programming model
  - Top down: replacing coupled driver
  - Bottom up: ocean, ice, land

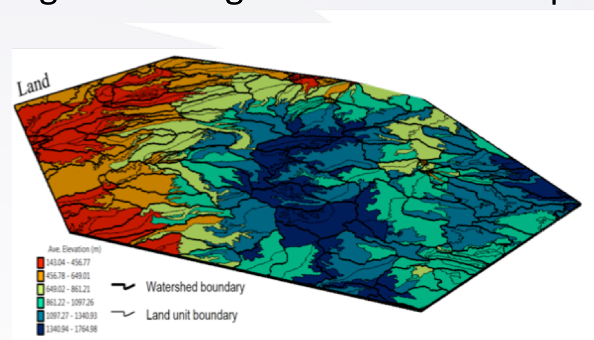
- ROO: Remapping Online-Offline
  - Remapping
    - Non-convex, irregular meshes
    - Adaptive and mesh-free remapping (support staggering on native pts)
    - Vector and property-preserving

- TIGGER: Time InteGration for Greater E3SM Robustness
  - Consistency and stability of the full integrated system

- Applications, mini-apps



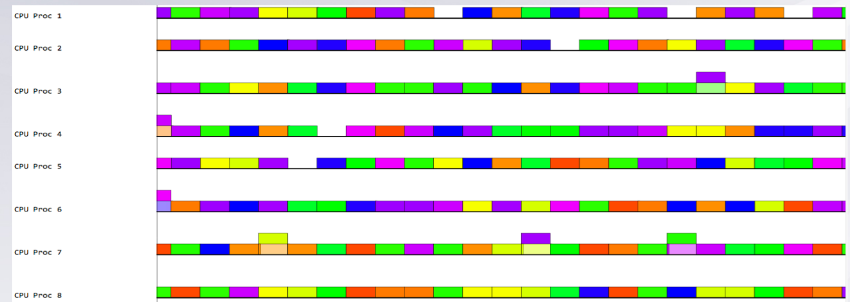
High-level diagram of land model processes



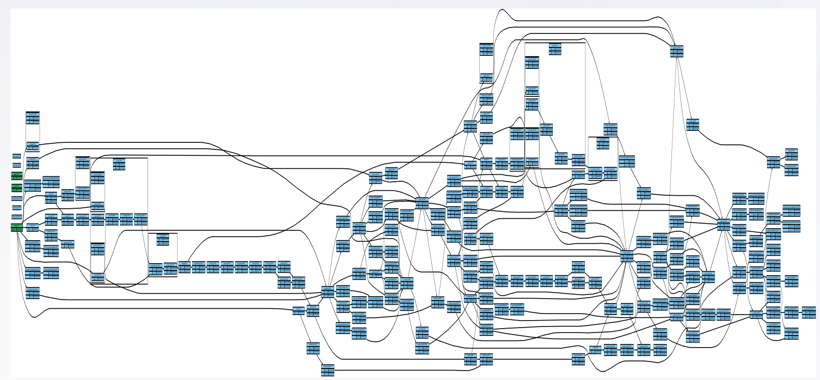
Remapping to land meshes with non-convex irregular geometry

CANGA: Task parallel status

- Created prototype coupled model
  - Dummy components in Regent
- Ocean and land prototypes
  - Ocean interfaces prototyped
  - Automated land kernel extraction
- Adopting FleCSI FleCSI
  - Run time abstraction layer supports Legion, Charm++, HPX, MPI
  - Includes control, execution and data models
  - Specialization layer (MPAS)
- Evaluation: too early
  - May work better for land, column physics, less clear on dycore
  - Still requires optimized version of tasks



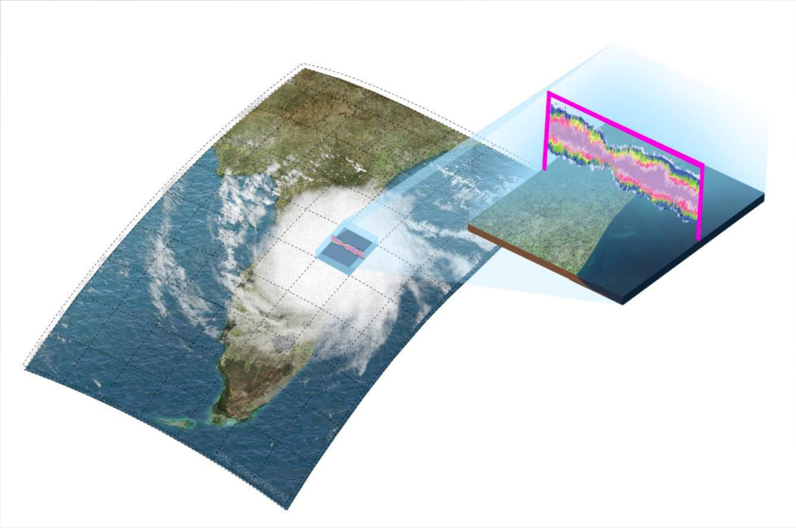
Parallel Regent prototype of coupled model



Initial task dependency graph from the MPAS-Ocean prototype code

The regional strategy

- Hierarchical model for machine hierarchy
- Use GPU/accelerator for subgrid ensembles
  - MMF/superparameterization
  - Ocean LES vertical mixing
- Provides substantial work for accelerator, minimize data transfer
- Potentially quicker approach to cloud-resolving scales
- Exascale Computing Program project for E3SM
  - Porting SAM/MMF to GPU on Summit



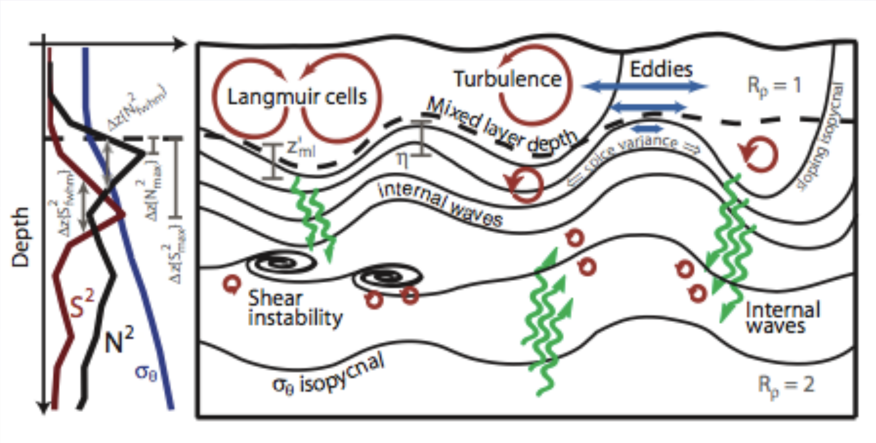


Figure credit: TM Shaun Johnston, SIO

Other approaches

- 3rd-party disruption: Domain specific language (DSL)
  - Fortran
    - Another aside: language issues…
  - Too many different patterns?
  - What level of abstraction?
- Collusion
  - Meeting with Chinese in NCAR Tower
  - 100s of coders, submit Gordon Bell app
- Ensembles
- New algorithms
  - Still where the biggest gains are found
  - Long time scales to build up constituency

Conclusions and Strategy

- Will not provide an endorsement to any candidate at this time
- No option will remove the need for significant performance tuning/optimization, better algorithms
- Task parallelism at all levels is required
  - Not enough parallelism in data decomposition alone
- Strategery
  - Continue to prototype programming model approaches (Kokkos, Legion/FleCSI)
  - Continuing ECP MMF/superparameterization approach to provide meatier kernels and an alternative path to cloud resolving scales
  - Develop hierarchical/hybrid approach
    - Continue porting with MPI+X versions to find right granularity and optimization strategy at each level