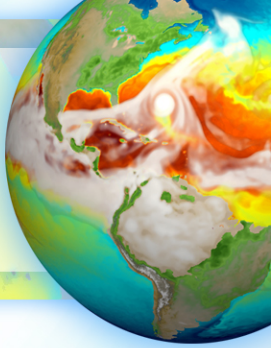
Massively Parallel Ultra-scale E3SM Land Model Development on Summit

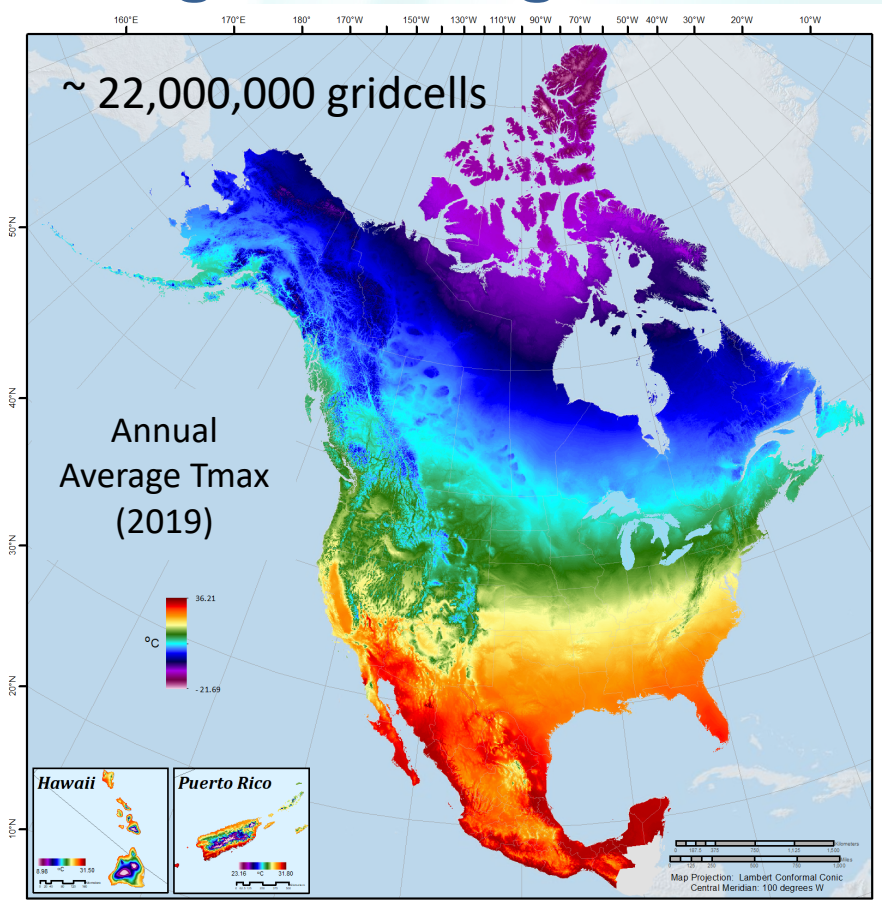


Peter Schwartz, Dali Wang, Fengming Yuan, Peter Thornton

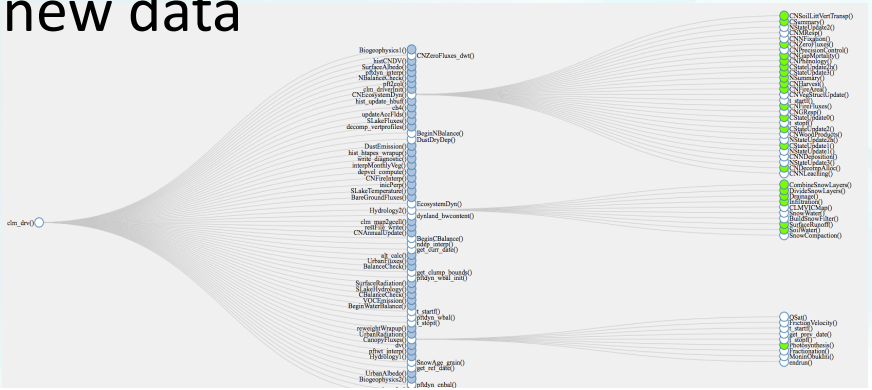
Marcia Branstetter, Rupesh Shrestha, Shih-Chieh Kao, Michelle Thornton

Oak Ridge National Laboratory

Target: 1km² grid resolution over N. America



Major efforts: New code and new data

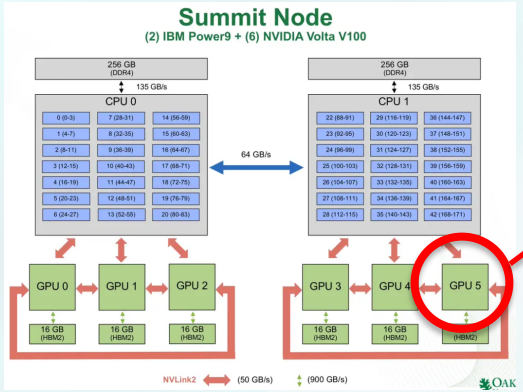


High-resolution surface weather inputs:

- Updated with new station data for 1980-2019
- Corrected time-of-observation biases in daily temperature
- Corrected temperature sensor biases in SNOTEL data record
- Applied temporal downscaling based on GSWP3

Computational Platform: GPUs on Summit

4,608 nodes, 27,000 NVIDIA Volta GPUs

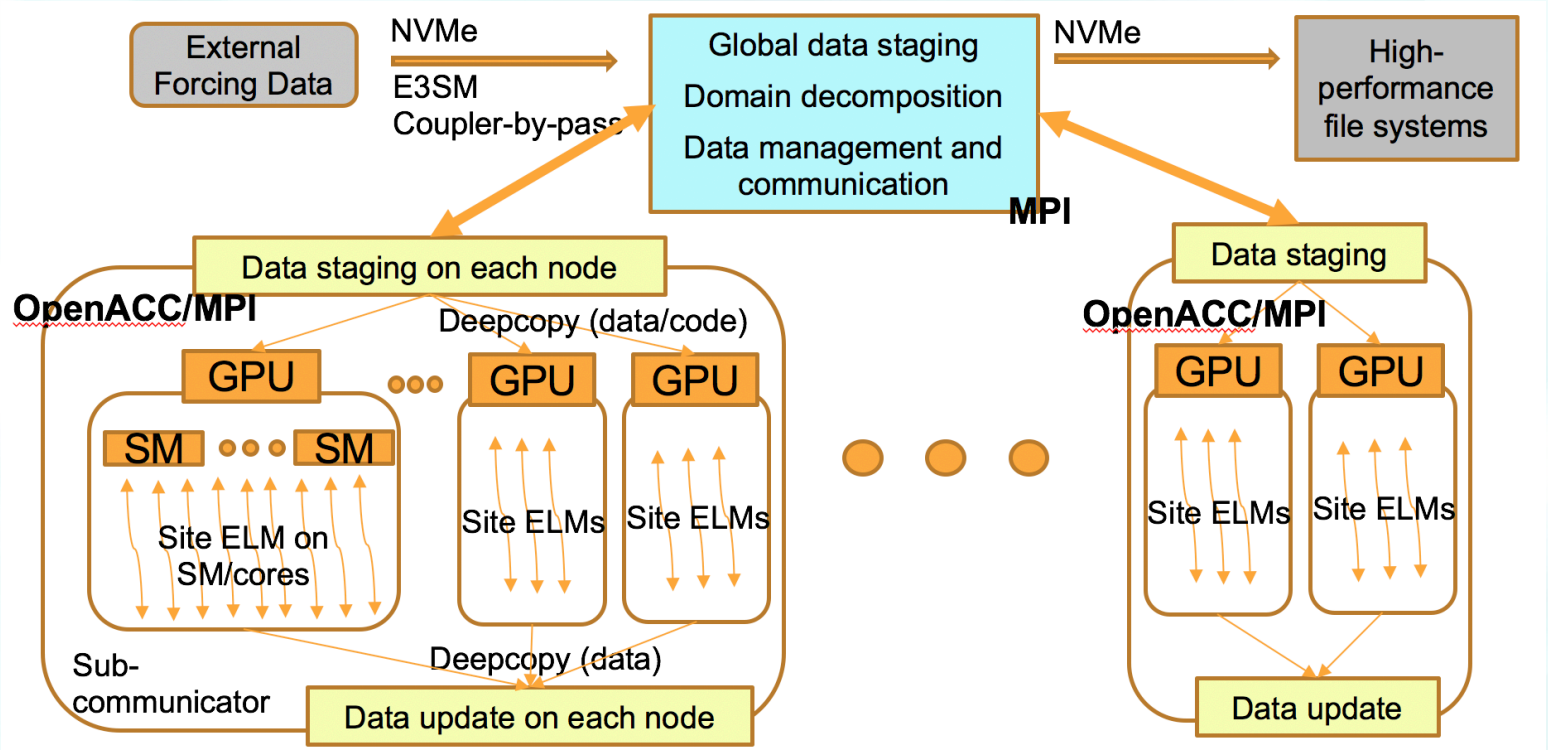


Each Summit node has 6 NVIDIA Volta V100 GPUs. We plan to have 1 ELM MPI task per GPU, so 6 MPI tasks per node

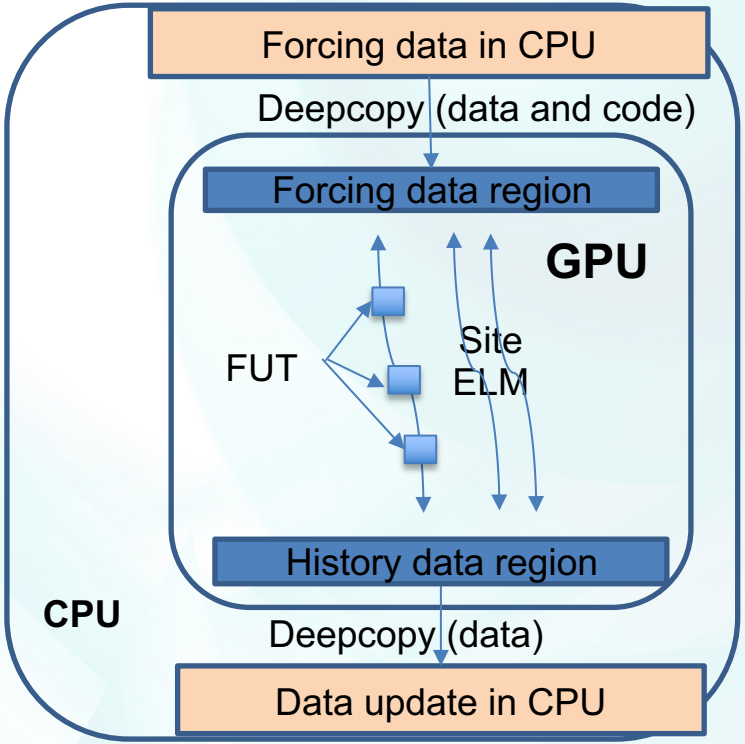
Each GPU has 80+ Streaming Multiprocessors (SMs) and 16 GB of shared memory (HBM2)

Our approach is to use the existing “clumps” parallelism in ELM (traditionally connected to OpenMP), and tie it to the double precision cores on the V100 GPUs via OpenACC, using **2 gridcell per clump**.

Parallel Strategy and Data Management



Implementation on Single GPU with Clumps



- Using OpenACC deep copy method to move entire ELM data structure (all clumps) onto GPU.
- Sequential execution of major science routines in ELM run step on GPU (6 clump-parallel regions).
- Update CPU on exit from GPU block, to allow I/O
- Using the **Functional Unit Testing** framework developed to rapidly prototype the GPU kernel
- Performance tests showed **near-ideal scaling** of compute time out to ~20 K clumps per GPU.
- As expected, data transfer time scales with number of clumps per GPU.

Data Structure Refactoring

- dealing with limited PGI/OpenACC functions

- Deepcopy Implementation:
  - PGI does not support all all derived types in ELM
  - Use pointer elements— even for types that held parameter — to ensure data is copied.
- Limited Use of Unstructured Data Regions
  - Only at beginning and end of run and history tapes when needed.
- Memory Requirements:
  - Each Site uses ≈3.2 MB for global variables.
  - OpenACC kernels requires ≈ 9.9 GB of memory
  - Current Limit is ≈2000 sites per GPU on SUMMIT.
  - Refactoring and optimizing local variables decreases the kernel memory and allows more sites.

Directive must accompany variable declaration

```
type(DecompCNParamsType) :: DecompCNParamsInst
!$acc declare create(DecompCNParamsInst)
```

Changing to pointers Example

```
type, public :: DecompCNParamsType
        real(r8), pointer :: cn_s1_cn => null()
        real(r8), pointer :: cn_s2_cn => null()
```

Subroutine Refactoring

- dealing with massive ELM functions

- Simplify complex subroutines
  - Removed custom timing, error checking, and I/O functions.
  - Use local pointer for certain derived data types
  - Break Class methods into several separate subroutines
- Automatic Batch Conversion of ELM Subroutines with Python Scripts
  - Recursively add acc directives
  - Identify class methods to streamline
  - Add new modules/developments
- Increase Data Parallelism
  - Introduced clump parallelism to certain functions: history and accumulation buffer update.

Class Method example

```
!call col_nf%Summary(bounds, num_soilc, filter_soilc)
call colnf_summary_acc(col_nf,bounds, num_soilc,
filter_soilc, dt)
```

Array slicing (error due to implicit intrinsic)

```
!call c2g(bounds,  col_cf%nee(begc:endc), &
!    lnd2atm_vars%nee_grc(begg:endg), &
!    c2l_scale_type= unity, l2g_scale_type=unity)
call c2g(bounds, nee(begc:endc) , nee_grc(begg:endg) , &
    c2l_scale_type= unity, l2g_scale_type=unity)
```

Adopted Performance Optimization Techniques

- Reduce / remove dynamic memory allocation
- Replace local arrays with scalars if possible (highest priority)
- Be cautious on memory allocation using filters, instead of entire clump.
- Increase data parallelism for routines that are most compute intensive (e.g., history buffer is parallelized in across fields as well as gridcells)
- Increase task parallelism and data locality for routines with hundreds of global memory accesses.
- Fine tuning of OpenACC parameters: gangs, registers, etc..

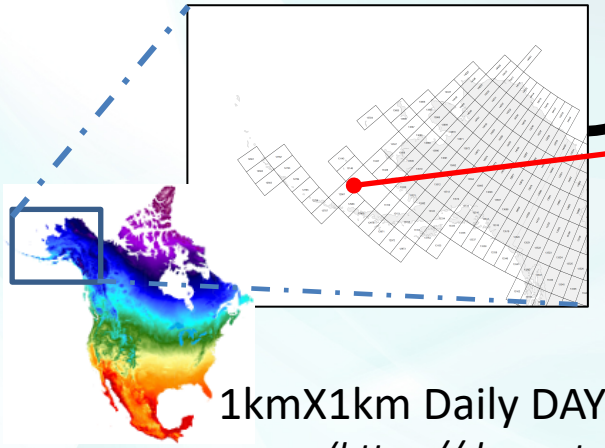
Timings (in seconds) for 3 sections of ELM code before and after first optimizations.

1890 Sites	Initial	Optimized
Pre_Science	65.3 s	27.0 s
BGC_Science	260 s	77.0 s
History	1,350,000 s	6.30 s

Current Efforts on Forcing Data Preparation

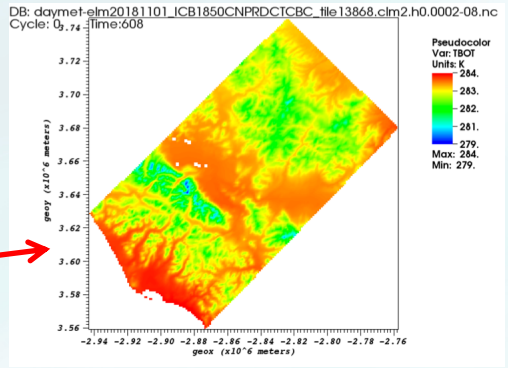
0.5ºX0.5º Three-Hourly GSWP3 ELM Atmospheric Forcing

+

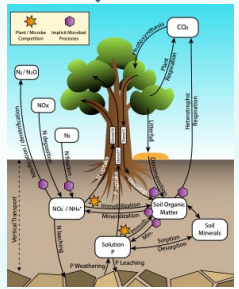


1kmX1km Daily DAYMET Forcing
(https://daymet.ornl.gov)

1kmX1km Sub-daily Atm. Forcing



High Spatial Res. Vegetation Maps

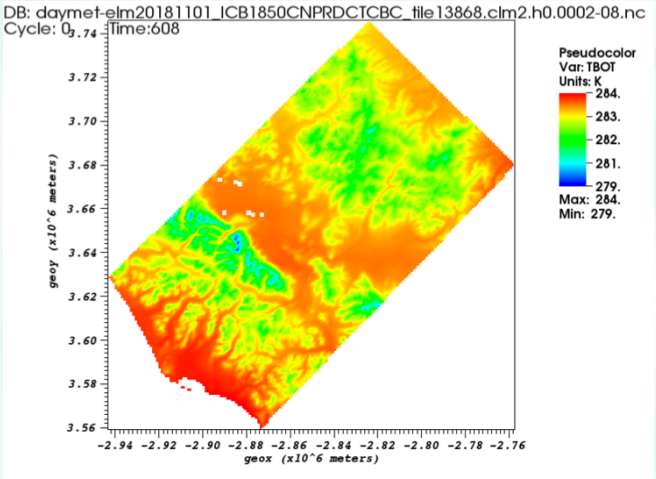


ELM

High Spatial Res. Soil (e.g. texture, SOM) Maps

Data Staging via Coupler-bypass

- Generate standalone GSWP3-Daymet4 dataset for North America
  - 1014 tiles (2.5 TB)
  - 23 - 38k gridcells per tile
  - 2.1-3.5 GB per tile
- Modify the OLMT coupler-bypass
  - Remove the built-in data generation function
- Modify the lnd_import_export function
  - Read in the GSWPS-Daymet4 dataset directly



Estimation on Computing Resource and Timing

- GPU memory constraints (2000 gridcells/GPU or 12K gridcells/node)
- 22M gridcells in NA requires 1800 Summit nodes (out of 4800)
- Goal: 4000 gridcell per GPU device

- Basic reference time (simulation over one tile with 5 cluster nodes)
  - 700-year simulation takes 2 weeks

- Current GPU timing is 125 seconds per model day / 64 minutes per month
- Goals: 1) 4 minutes per month (16x) / 700-year simulation in 24 days
  2) 1 minutes per month (extra 4x) / 700-year simulation in 6 days