https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

Parallel SnowModel (v1.0): a parallel implementation of a Distributed Snow-Evolution Modeling System (SnowModel)

Ross Mower[1,2], Ethan D. Gutmann[1], Jessica Lundquist[2], Glen E. Liston[3], Soren Rasmussen[1]

[1]The National Center for Atmospheric Research, Boulder, Colorado, USA
[2]Department of Civil and Environmental Engineering, University of Washington, Seattle, Washington, USA
[3]Cooperative Institute for Research in the Atmosphere, Colorado State University, Fort Collins, Colorado, USA

Correspondence to: Ross Mower (rossamower@ucar.edu)

Abstract. SnowModel, a spatially distributed, snow-evolution modeling system, was parallelized using Coarray Fortran for high-performance computing architectures to allow high-resolution (1 m to 100's of meters) simulations over large, regional to continental scale, domains. In the parallel algorithm, the model domain is split into smaller rectangular sub-domains that are distributed over multiple processor cores using one-dimensional decomposition. All of the memory allocations from the original code have been reduced to the size of the local sub-domains, allowing each core to perform fewer computations and requiring less memory for each process. A majority of the subroutines in SnowModel were simple to parallelize; however, there were certain physical processes, including blowing snow redistribution and components within the solar radiation and wind models, that required non-trivial parallelization using halo-exchange patterns. To validate the parallel algorithm and assess parallel scaling characteristics, high-resolution (100 m grid) simulations were performed over several western United States domains and over the contiguous United States (CONUS). The CONUS scaling experiment had approximately 71% parallel efficiency; runtime decreased by a factor of 32 running on 2304 cores relative to 52 cores (the minimum number of cores that could be used to run such a large domain as a result of memory and time limitations). CONUS 100 m simulations were performed for 21 years (2000 – 2021) using 46,238 and 28,260 grid cells in the *x* and *y* dimensions, respectively. Each year was simulated using 1800 cores and took approximately 5 hours to run.

1 Introduction

The cryosphere (snow and ice) is an essential component of Arctic, mountain, and downstream ecosystems, Earth's surface energy balance, and freshwater resource storage (Huss et al., 2017). Globally, half the world's population depends on snowmelt (Beniston, 2003). In snow-dominated regions like the Western United States, snowmelt contributes to approximately 70% of the total annual water supply (Foster et al., 2011). In these regions, late-season streamflow is dependent on the deepest snow drifts and therefore longest-lasting snow (Pflug and Lundquist, 2020). Since modeling snow-fed streamflow accurately is largely dependent on our ability to predict snow quantities and the associated spatial and temporal variability (Clark and Hay, 2004), high-temporal and -spatial resolution snow datasets are important for predicting flood hazards and managing freshwater resources (Immerzeel et al., 2020).

https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

The spatial and temporal seasonal snow characteristics also have significant implications outside of water resources. Changes in fractional snow-covered area affect albedo and thus atmospheric dynamics (Liston, 2004; Liston and Hall, 1995). Avalanches pose safety hazards to both transportation and recreational activities in mountainous terrain; the prediction of which requires high-resolution (meters) snow datasets (Morin et al., 2020; Richter et al., 2021). Additionally, the timing and duration of snow-covered landscapes strongly influence how species adapt, migrate, and survive (Boelman et al., 2019; Liston et al., 2016; Mahoney et al., 2018).

To date, the primary modes for estimating snow properties and storage have come from observation networks, satellite-based observations, and physically derived snow algorithms in land surface models (LSMs). However, despite the importance of regional, continental, and global snow, estimates of snow properties over these scales remain uncertain, especially in alpine regions where wind, snow, and topography interact (Boelman et al., 2019; Dozier et al., 2016; Mudryk et al., 2015). Observation datasets used for spatial interpolation of snow properties and forcing datasets used in LSMs are often too sparse in mountainous terrain to accurately resolve snow spatial heterogeneities (Dozier et al., 2016; Renwick, 2014). Additionally, remotely sensed products have shown deficiencies in measuring snowfall rate (Skofronick-Jackson et al., 2013), snow-water equivalent (SWE), and snow depth (Nolin, 2010), especially in mountainous terrain where conditions of deep snow, wet snow, and/or dense vegetation may be present (Lettenmaier et al., 2015; Takala et al., 2011; Vuyovich et al., 2014). However, LSMs using high-resolution inputs, including forcing datasets from regional climate models (RCMs), have demonstrated realistic spatial distributions of snow properties (Wrzesien et al., 2018).

For several decades, a distributed snow-evolution modeling system (SnowModel) has been developed, enhanced, and tested to accurately simulate snow properties across a wide range of landscapes, climates, and conditions (Liston and Elder, 2006b; Liston et al., 2020). To date, SnowModel has been used in over 200 refereed journal publications; a short listing of these is provided by Liston et al. (2020). Models like SnowModel can be computationally expensive. In these models, the required computational power increases with the number of grid cells covering the simulation domain. Finer grid resolutions usually imply more grid cells and higher accuracy resulting from improved representation of process physics at higher resolutions. The original serial SnowModel code was written in Fortran 77 and could not be executed in parallel using multiple processor cores. As a result, SnowModel's spatial and temporal simulation domains (number of grid cells and time steps) were previously limited by the speed of one core and the memory available on the single-computer. Note that a "processor" refers to a single central processing unit (CPU) and typically consists of multiple cores, each core is able to run one or more processes in parallel.

Recent advancements in multiprocessor computer technologies and architectures have allowed for increased performance in simulating complex natural systems at high resolutions. Parallel computing has been used on many LSMs to reduce compute time and allow for higher accuracy results from finer grid simulations (Hamman et al., 2018; Miller et al., 2014; Sharma et al., 2004). Our goal was to develop a parallel version of SnowModel (Parallel SnowModel) using Coarray Fortran (CAF) syntax without making significant changes to the original SnowModel code physics or structure. CAF is a Partitioned Global

https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

Address Space (PGAS) programming model and has been used to run atmospheric models on 100,000 cores (Rouson et al., 2017).

In parallelizing numerical models, a common strategy is to decompose the domain into smaller sub-domains that get distributed across multiple processes (Dennis, 2007; Hamman et al., 2018). For rectangular gridded domains (like SnowModel), this preserves the original structure of the spatial loops and utilizes direct referencing of neighboring grids (Perezhogin et al., 2021). The parallelization of many LSMs involve "embarrassingly parallel" problems requiring minimal to no processor communication (Parhami, 1995); in this case, adjacent grid cells do not communicate with each other (an example of this would be where each grid cell represents a point, or one-dimension, snowpack model that is not influenced by nearby grid cells).

While much of the SnowModel's logic can be considered "embarrassingly parallel", SnowModel also contains "non-trivial" algorithms within the solar radiation, wind, and snow redistribution models. Calculations within these algorithms often require information from neighboring grid cells, either for spatial derivative calculations or for horizontal fluxes of mass (e.g., saltating or turbulent-suspended snow) across the domain. Therefore, non-trivial parallelization requires implementing algorithm changes that allow computer processes to communicate and exchange data. The novelty of the work presented here includes 1) the presentation of Parallel SnowModel and high-resolution (100 m) distributed snow datasets over CONUS; 2) demonstrating how a simplified parallelization approach using CAF and one-dimensional decomposition can be implemented in geoscientific algorithms to scale over large domains; and 3) demonstrating an approach for non-trivial parallelization algorithms that involve spatial derivatives and fluxes using halo-exchange (HX) techniques.

In Sect. 2, we describe SnowModel and provide a motivation for its parallelization. In Sect. 3, we explain our parallelization approach using CAF and the module developed that partitions the two-dimensional domain in the *y* dimension and organizes the non-trivial communication necessary to produce accurate results. In Sect. 4, we validate results from Parallel SnowModel compared to serial simulations, discuss the evolution of the performance of the parallel algorithm, analyze the efficiency of Parallel SnowModel using strong scaling metrics over several basins throughout the United States, and present Parallel SnowModel results over CONUS. Lastly, end with a discussion in Sect. 5 and a conclusion in Sect. 6.

2 SnowModel

SnowModel is a spatially distributed snow-evolution modeling system designed to model snow properties (e.g., snow depth, SWE, snow melt, snow density) over different landscapes and climates (Liston and Elder, 2006b). The most complete and up-to-date description of SnowModel can be found in the Appendices of Liston et al. (2020). While many snow modeling systems exist, SnowModel is standing to benefit from parallelization across larger domains because of its ability to input high-resolution meteorological data (e.g., wind, radiation, and precipitation) and model fine-scale snow properties and redistribution processes. SnowModel is designed to simulate domains on a structured grid with spatial resolutions ranging from 1 to 200 m (although it has the ability to simulate coarser resolutions, as well) and temporal resolutions ranging from

https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

10 m to 1 d. The primary modeled processes include accumulation from frozen precipitation; blowing-snow redistribution and sublimation; interception, unloading, and sublimation within forest canopies; snow-density and grain-size evolution; and snowpack ripening and melt. These processes are distributed into four interacting submodules: MicroMet defines the meteorological forcing conditions (Liston and Elder, 2006a), EnBal describes surface and energy exchanges (Liston, 1995; Liston et al., 1999). SnowPack is a multilayer snowpack sub-model that simulates the evolution of snow properties and the moisture and energy transfers between layers (Liston and Hall, 1995; Liston and Mernild, 2012), and SnowTran-3D calculates snow redistribution by wind (Liston et al., 2007). Additionally, the initialization submodules that read in the model parameters, distribute inputs across the modeled grid, allocate arrays, etc., include PreProcess and ReadParam. SnowModel incorporates first-order physics required to simulate snow evolution within each of the global snow classes [e.g., Ice, Tundra, Boreal Forest, Montane Forest, Prairie, Maritime, and Ephemeral; (Sturm and Liston, 2021; Liston and Sturm, 2021).

The required inputs for SnowModel include 1) temporally varying meteorological variables of precipitation, wind speed and direction, air temperature, and relative humidity taken from meteorological stations or atmospheric models and 2) spatially distributed topography and land-cover type (Liston & Elder, 2006a). The following inputs were used for the experiments conducted in Sect. 4: USGS National Elevation Dataset (NED) for topography (Dean B. Gesch, 2018), The North American Land Change Monitoring System (NALCMS) Land Cover 2015 map for vegetation (Homer et al., 2015; Jin et al., 2019; Latifovic et al., 2016), and forcing variables from either the North American Land Data Assimilation System (NLDAS-2) (Mitchell, 2004; Xia, 2012a, b) on a 1/8 degree (approximately 12 km) grid or a high-resolution Weather Research Forecast (WRF) model from the National Center for Atmospheric Research (NCAR) on approximately a 4 km grid (Rasmussen et al., 2023). The high-performance computing architectures used include NCAR's Cheyenne supercomputer, which is a 5.43-petaflop SGI ICE XA Cluster featuring 145,152 Intel Xeon processes in 4,032 dual-socket nodes and 313 TB of total memory (Laboratory, 2019) and The National Aeronautics and Space Administration's (NASA) Center for Climate Simulation (NCCS) Discover supercomputer with a 1,560-teraflop SuperMicro Cluster featuring 20,800 Intel Xeon Skylake processes in 520 dual-socket nodes and 99.84 TB of total memory (Carriere, 2023).

2.1 Parallelization Motivation

The answers to current snow science, remote sensing, and water management questions require high-resolution data that covers large spatial and temporal domains. While modeling systems like SnowModel can be used to help provide these datasets, running them on single-processor workstations imposes limits on the spatiotemporal extents of the produced information. Serial simulations are limited by both execution time and memory requirements, where the memory limitation is largely dependent on the size of the simulation domain. Up to the equivalent of 175 two-dimensional and 10 three-dimensional arrays are held in memory during a SnowModel simulation, depending on the model configuration. In analyzing the performance of the parallel algorithm (Sect. 4), serial simulations were attempted over six domains throughout the United States at 100 m grid resolution. The spatial location, domain dimensions (e.g., number of grids in the x and y

https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

dimensions), and memory requirements, derived from the peak_memusage package (https://github.com/NCAR/peak_memusage), for the simulation experiments are highlighted in Fig. 1. The simulations were executed on Cheyenne for 16 timesteps on 23-24 March 2018 using NLDAS-2 forcing. The three largest domains (Pacific Northwest (PNW), Western U.S. and CONUS) could not be executed in serial using Cheyenne's normal or large memory (55 GB and 109 GB, respectively) compute nodes due to both exceedances of the 12 h wall-clock limit and memory availability. Furthermore, we estimate that using a currently available, state of the art, single-processor workstation, would require approximately 120 d of computer time to perform a 1 y model simulation over the CONUS domain (Fig. 1) at a 100 m grid increment and a 3 h time step. SnowModel is regularly used to perform multi-decade simulations, for trend analyses, climate change studies, and retrospective analyses (Liston and Hiemstra, 2011; Liston et al., 2020; Liston et al., 2022). If this 1 y, 100 m, CONUS domain was simulated for a 40 y period (e.g., 1980 through present), it would take approximately 4800 d, or over 13 y, of computer time. Clearly such simulations are not practical using single-processor computer hardware and software algorithms.

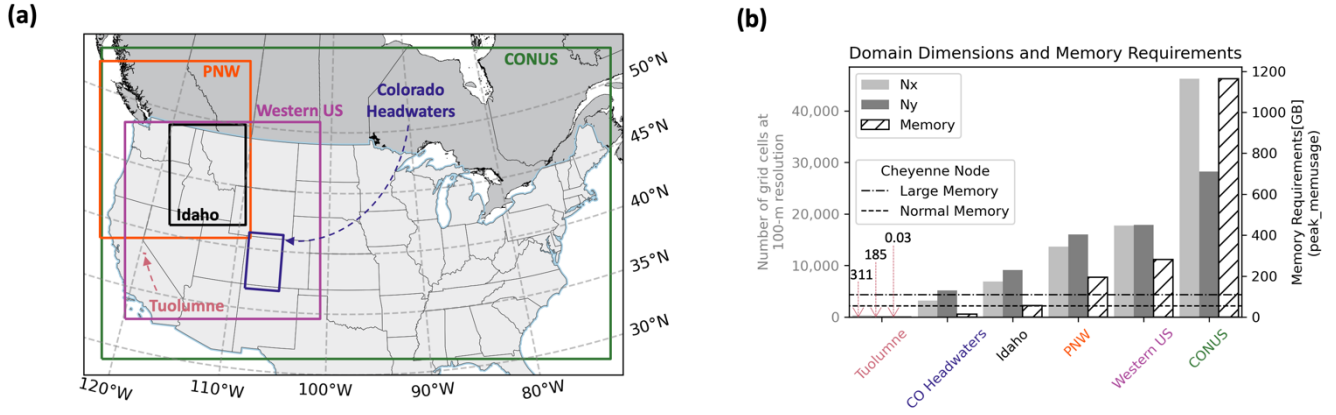

Figure 1: ***Spatial location of simulated domains on WRF's lambert conformal projection (Rasmussen et al., 2023) (a) and corresponding grid dimensions (Nx – number of grids in x dimension; Ny – number of grids in y dimension) and memory obtained from peak_memusage package required for single-layer SnowModel simulation experiments (b). For reference, the dashed lines represent the normal and large memory thresholds (55 and 109 GB) for Cheyenne's SGI ICE XA cluster.***

3 Parallel Approach

In parallelizing SnowModel and distributing computations and memory over multiple processes, we hope to be able to run regional to continental sized simulations efficiently. Some of the model configurations were not parallelized for reasons including ongoing development in the serial code base and limitations to the parallelization approach. These configurations are further discussed in Appendix A. This section introduces the syntax and framework used to parallelize SnowModel, including the partitioning strategy, algorithms involving non-trivial processor communication via halo exchange, and file input and output (I/O).

https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

3.1 Fortran Coarrays

CAF, formerly known as F-, (Iso/Iec, 2010; Numrich and Reid, 1998; Numrich et al., 1997) is the parallel language feature of Fortran that was used to parallelize SnowModel. CAF is similar to Message Passing Interface (MPI) libraries in that it uses the Single Program Multiple Data (SPMD) model where multiple independent cores simultaneously execute a program. SPMD allows for distributed memory allocation and remote memory transfer. However, unlike MPI, CAF uses the PGAS parallel programming model to handle the distribution of computational tasks amongst processes (Coarfa et al., 2005). In the PGAS model, each process contains local memory that can be accessed directly by all other processes. While CAF and MPI syntax often refers to processes as images or ranks, for consistency, we will continue to use the term "process". Ultimately, CAF offers a high-level syntax that exploits locality and scales effectively (Coarfa et al., 2005). For simulation comparisons, we used OpenCoarrays, a library implementation of CAF (Fanfarillo et al., 2014) utilized by the gfortran compiler; intel and cray compilers both have independent CAF implementations.

Upon initiation of a CAF program, the number of processes is designated and replicates the program N_p times, with each process allocating and storing its own memory locally. Local arrays contain information specific to that process's local domain, while coarrays are data structures used to communicate information among multiple processes. CAF syntax uses square brackets as subscripts to allocate coarrays and transfer data from one process to another. A variable without square brackets refers to the current process's copy of a coarray variable. As an example, the coarray logic in Algorithm 1 (Fig. 2) demonstrates a CAF program executed with three processes. In this program memory from the first element of a one-dimensional array, U, residing on the second and third processes (P_2 and P_3) gets copied into the fifth element of the local array U on the second and first processes (P_2 and P_1), respectively. In the code logic and hereafter, *proc* represents each process's identification number, while N_p represents the total number of processes used to execute a program. Both integers are created through intrinsic CAF functions (e.g., *this_image()* and *num_image()*, respectively).

Algorithm 1 : CAF code syntax

```
program caf_test
  implicit none
  integer :: U(5)[*] !allocate coarray U
  integer :: me  !variable for process identification
  integer :: Np !variable for total number of processes
  proc = this_image()  !coarray function to identify process number
  Np = num_images() !coarray function to identify total number of processes
  U = [1, 2, 3, 4, 5] !step 1: initialize U
  if (me /= 1) U(5) [proc-1] = U(1) !step 2: transfer data
end program
```

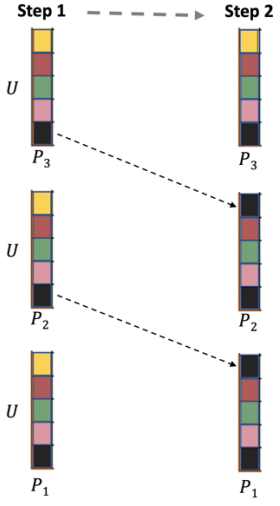

Figure 2: Example logic and schematic transferring data and updating array values using coarrays.

https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

3.2 Partitioning Algorithm

The partitioning strategy identifies how the workload gets distributed amongst processes in a parallel algorithm. Both the data structures and physical processes involved in SnowModel justify a one-dimensional decomposition strategy in the y dimension. The multidimensional arrays of SnowModel are stored in row-major order, meaning that the x dimension is contiguous in memory. Additionally, dominant wind directions and therefore predominant snow redistribution occurs in the east-west direction as opposed to south-north directions. The partitioning algorithm decomposes the domain in the y dimension, and allocates local arrays based on N_p. For domain decomposition, the computational global domain N_x x N_y is separated into N_x x l_ny blocks. If N_y is evenly divisible by N_p, then $l\_ny_p = \frac{N_y}{N_p}$. If integer division is not possible, then the remaining rows are distributed evenly amongst the processes starting at the bottom of the computational domain. Figure 3 demonstrates how a serial domain containing 10 grid cells in the x and y dimensions would be decomposed with four processes using our partitioning strategy. The domain decomposition over several processes requires mapping information across local entities. Two arrays are created that identify the l_ny_p for each processor (*partition_ny*) and the starting index of each processor's local domain within the context of the global domain (*prefix_sum*). These arrays are used for indexing purposes during file I/O and processor communication.

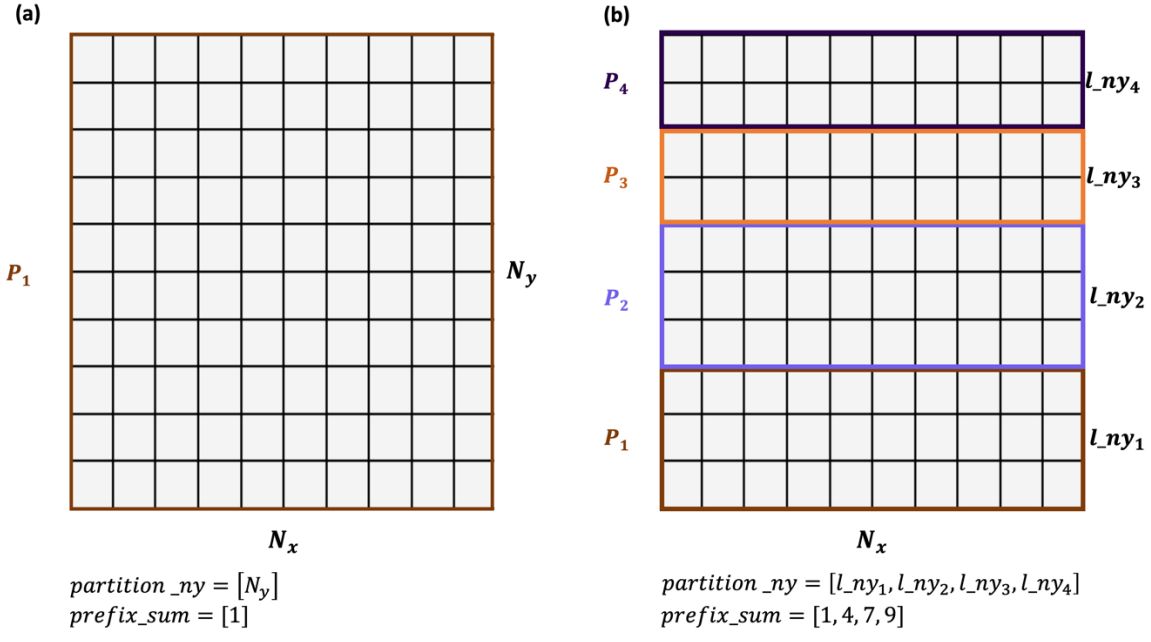

Figure 3: ***Example 10 x 10 global domain and partitioning for serial simulation (a), and parallel simulation using four processes (b).***

3.3 Non-trivial Parallelization

Each process has sufficient information to correctly execute a majority of the physical computations within SnowModel. However, there are certain subroutines where grid computations require information from neighboring grid cells (e.g., data

https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

dependencies) and therefore information outside of a process's local domain. For SnowModel, these subroutines typically involve the transfer of blowing snow or calculations requiring spatial derivatives. Furthermore, with our one-dimensional decomposition approach, each grid cell within a process's local domain has sufficient information from its neighboring grid cells in the *x* dimension but potentially lacks information from neighboring grid cells in the *y* dimension. As a regular grid method, SnowModel lends itself to process communication via HX, where coarrays are used in remote calls. HX using CAF involves copying boundary data into coarrays on neighboring images and using information from the coarrays to complete computations (Fig. 4). Although the entire local array could be declared a coarray and accessed by remote processes more directly, some CAF implementations impose additional constraints upon coarray memory allocations that can be problematic for such large allocations.

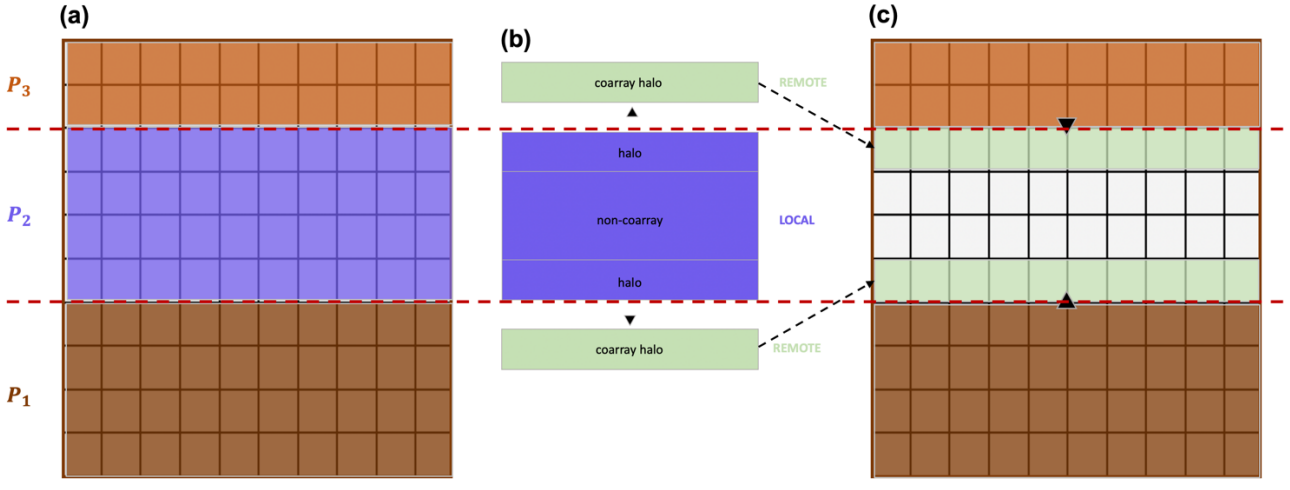

Figure 4: ***Schematic showing HX using coarrays. The steps include: initial gridded representation of local arrays for three processes (a), P_2 copying boundary data into coarrays for remote access (b), neighboring processes (P_1 and P_3) stitching coarray to local domains (c).***

3.3.1 Topography – Wind and Solar Radiation Models

The wind and solar radiation models in MicroMet require information about surrounding surface topography. The wind model requires surface curvature, and the solar radiation model requires surface slope and aspect. These vary at each timestep as snow accumulates and melts because the defined surface includes the snow surface on top of the landscape. The curvature (Ω_c), for example, is computed at each model grid cell using the spatial gradient of the topographic elevation of eight neighboring grid cells ($Z_w, Z_E, Z_S, Z_N, Z_{SW}, Z_{NE}, Z_{NW}, Z_{SE}$ $[m]$, where Z_w corresponds to the elevation of the grid cell west of the current grid cell, Z_{NW} is the elevation of the grid cell northwest of the current grid cell, etc.) and a curvature length scale or radius, η $[m]$, which is a pre-defined parameter equal to approximately half the wavelength of the topographic features within a domain (Eq. 1) (Liston and Elder, 2006b).

$$\Omega_c = \frac{1}{4}\left[\frac{z-\frac{1}{2}(z_W+z_E)}{2\eta} + \frac{z-\frac{1}{2}(z_S+z_N)}{2\eta} + \frac{z-\frac{1}{2}(z_{SW}+z_{NE})}{2\sqrt{2\eta}}\,\frac{z-\frac{1}{2}(z_{NW}+z_{SE})}{2\sqrt{2\eta}}\right], \quad (1)$$

https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

Using the parallelization approach discussed above, processes lack sufficient information to make curvature calculations for the bordering grid cells along the top and/or bottom row(s) within their local domains. For example, all grid cells along the top row of P_1 will be missing information from nearby grid cells to the north (Z_{NW}, Z_N, and Z_{NE}), and require topographic elevation ($topo$) information from the bottom row(s) of the local domain of P_2 to make the calculation (Algorithm 2, Fig. 5a). HX is performed to distribute row(s) (inc) of $topo$ data to each process that is missing that information in their local domains (Fig. 5b). Processes whose local domains are positioned in the bottom or top of the global domain will only perform one HX with their interior neighbor, while interior processes will perform two HXs. By combining and appropriately indexing information from the process's local array and received coarrays ($coarray\_n$ and $coarray\_s$) of topographic elevation, an accurate curvature calculation can be performed using this parallel approach (Fig. 5c). Lastly, *sync all* (Algorithm 2) is an intrinsic CAF function that synchronizes all of the processes. In other words, no process will go past this point in the algorithm until all of the processes have arrived. This is an important component of HX algorithms because it helps to prevent processes from making calculations before they have received important coarray information from a neighboring process. However, synchronization statements have an associated cost of decreasing the speed and efficiency of an algorithm and therefore should be minimized (discussed further in Sect. 4.2).

Algorithm 2 : Halo exchange

```
if (proc /= 0) coarray_s(1:nx,1:inc) [proc-1] = &
 &   topo(1:nx,1:inc) !send coarray to process below

if (proc /= Np-1) coarray_n(1:nx,1:inc) [proc+1] = &
 &   topo(1:nx, l_ny – (inc-1):l_ny) !send coarray to process above

sync all !synchronize processes
```

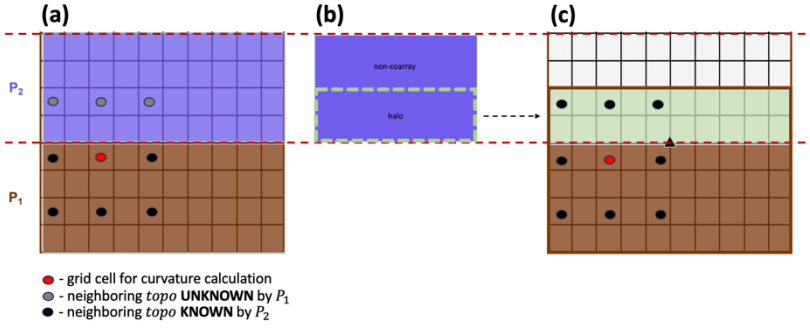

Figure 5: Schematic for HX of $coarray\_s$ used in the curvature calculation by P_1, where $inc = 2$. Prior to HX, P_1 contains insufficient information to perform the curvature calculation (a), grid cells (halo) within the local domain of P_2 (b) are transferred to P_1 via coarrays (coarray_s, Alg. 2) (c). At this point, P_1 has sufficient information to make the curvature calculation.

3.3.2 Snow Redistribution

Wind influences the mass balance of the snowpack by suspending and transporting snow particles in the air (turbulent-suspension) and by causing snow grains to bounce on top of the snow surface (saltation). Furthermore, field measurements at alpine sites in Colorado and Wyoming, have shown that snow can be transported up to 6 km due to saltation and suspension (Tabler, 1975). Snow redistribution in SnowTran-3D is defined using a mass balance equation describing the temporal variation of snow depth at a point [Eq. 2 (Liston and Sturm, 1998), see their Fig 2], where changes in the horizontal mass-transport rates of saltation, Q_s [kg m^{-1} s^{-1}], changes in turbulent-suspended snow, Q_t [kg m^{-1} s^{-1}], sublimation of transported snow particles, Q_v [kg m^{-1} s^{-1}], water-equivalent precipitation rate, P [m s^{-1}], and snow and water density, ρ_s [kg m-3], combine to describe the time rate of change of snow depth ζ [m]. At each timestep, snow redistribution ($d\zeta$) is solved for

https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

each grid cell through spatial derivatives ($\frac{d}{dx}$,$\frac{d}{dy}$) from neighboring grid cells. Since spatial derivatives and horizontal mass-transport rates of saltating and suspended snow are required, processor communication is also required along the boundary grid cells through HX.

$$\frac{d\zeta}{dt} = \frac{1}{\rho_s}\left[\rho_w P - \left(\frac{dQ_s}{dx} + \frac{dQ_t}{dx} + \frac{dQ_s}{dy} + \frac{dQ_t}{dy}\right) + Q_v\right], \quad (2)$$

In SnowModel, the saltation and suspension algorithms are separated into northerly, southerly, easterly, and westerly fluxes based on the *u* and *v* components of wind direction for each grid cell. Figure 6 shows a simplified serial algorithm and schematic for the saltation flux from a southerly wind. To start, SnowModel initializes the maximum saltation flux (*Qsalt_v*) as the boundary condition. To calculate the saltation flux, SnowModel iterates over continuous sections (*jstart* and *jend*) of the same wind direction, updates the change in saltation fluxes from upwind grid cells (*dQsalt*), and the change in saltation flux from the given wind direction (*dh_qsalt_v*), and makes adjustments of these fluxes based on the snow availability due to vegetation height and snow compaction (Liston and Elder, 2006b).

Algorithm 3 : Serial change in saltation flux (southerly winds)

```
initialize boundary condition (Qsalt_v)
do j = 1, Nx
   do j = jstart, jend
      calculate dQsalt and dh_salt_v
      update Qsalt_v and dh_salt_v ! based on vegetation height and snow compaction
   end do
end do
```

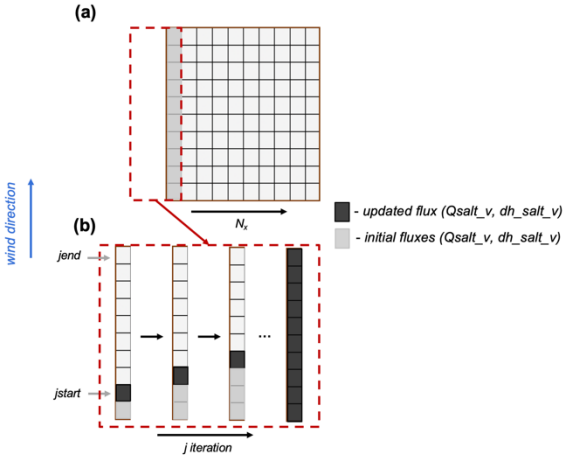

Figure 6: ***Schematic of serial algorithm showing change in saltation flux (Qsalt_v and dh_salt_v) due to southerly winds over domain (a), and the iteration to update the saltation fluxes Nx = 1 (b).***

Similar logic is used for the parallel implementation of saltation and suspension fluxes with an additional iteration (*salt_iter*) to update the boundary condition for each process via HX. This allows the fluxes to be communicated from one process's local domain to another. To prevent excessive iterations, *salt_iter* was provided a maximum bound that is equivalent to snow being transported 15 km via saltation or sublimation. This number was chosen based off prior field measurements (Tabler, 1975) and simulation experiments. It is possible that in other environments an even larger length may be required, to be guaranteed to match the serial results in all cases, the number of iterations would have to be equal to the number of processes; however, this would result in no parallel speed up and has no practical benefit. A skeleton of the parallel calculation of the change in saltation due to southerly winds is illustrated in Algorithm 4 in Fig. 7. In the parallel algorithm,

https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

the *bc_halo_exchange* subroutine performs a HX of *Qsalt_v* grid cells from upwind processes, allowing the saltation flux to be transported from one process's local domain to the next.

Algorithm 4 : Parallelized change in saltation flux (southerly winds)

```
initialize boundary condition (Qsalt_v)
do iter = 1, salt_iter
   bc_halo_exchange(Qsalt_v)
   do j = 1, Nx
      do j = jstart, jend
         calculate dQsalt and dh_salt_v
         update Qsalt_v and dh_salt_v ! based on vegetation height and snow compaction
      end do
   end do
end do
```

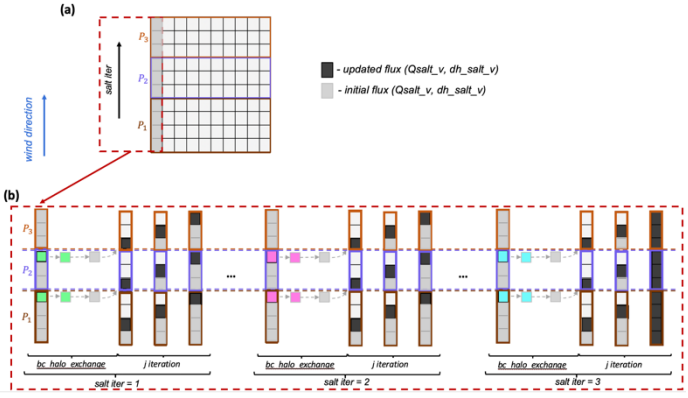

Figure 7: ***Schematic of the parallel algorithm showing change in saltation flux (Qsalt_v and dh_salt_v) due to southerly winds over a domain simulated with three processes (P1, P2, P3) (a), and over three saltation iterations (salt iter) for Nx = 1 (b). Before each iteration, the boundary condition of the saltation flux (bc_halo_exchange) gets updated using HX.***

3.4 File I/O

File I/O management can be a significant bottleneck in parallel applications. Parallel implementations that are less memory restricted commonly use local to global mapping strategies, or a ***centralized*** approach for file I/O (Fig. 8a). However, this approach requires that each process stores global arrays for input and output variables and creates a substantial bottleneck as the domain size scales (Sect. 4.2). To improve performance, ***distributed*** file I/O can be implemented, where input and output files are directly and concurrently assessed by each process (Fig. 8b).

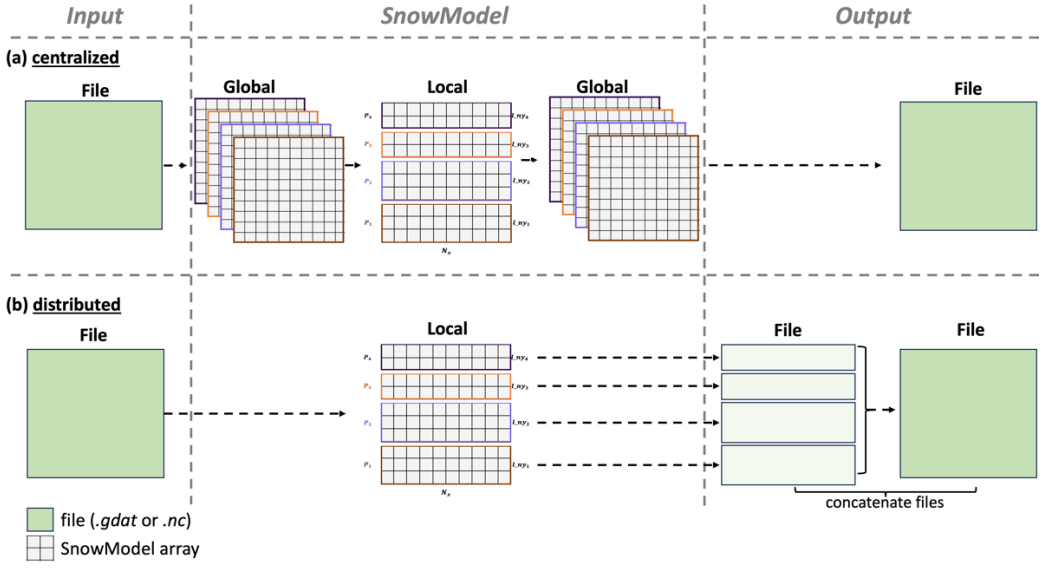

Figure 8: ***Schematic of global to local mapping for file I/O using a centralized approach with four processes (a), and distributed file I/O where each process reads and writes data corresponding to its local domain (b).***

https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

SnowModel contains static spatial inputs that do not vary over time, e.g., topography and land cover, and dynamic spatial inputs, e.g., air temperature and precipitation, that vary spatially and temporally. The static inputs are of a higher resolution compared to the dynamic inputs (cf., topography is on the model grid, while atmospheric forcing is almost always more widely spaced). In an attempt to balance performance and consistency with the serial logic of the code, we used a mixed parallel file I/O approach. File input (reading) is performed in a distributed way for the static inputs and in a centralized way for dynamic inputs, while file output (writing) is performed in a distributed way, as described further below.

3.4.1 Parallel Inputs

SnowModel's primary static spatial inputs include topography and vegetation data. However, depending on the simulation configuration, additional spatial inputs representing gridded values of latitude and longitude may be required. Acceptable static input file types include binary and ASCII files and can be read in a centralized or distributed manner. However, since the resolution of static inputs is identical to the resolution of the modeled grid, these arrays need to be read in a distributed manner. This is crucial for Parallel SnowModel's ability to scale to regional and continental sized domains, as storing a single copy of the full domain topography over CONUS would require 5.23 GB of memory. Therefore, if a node containing 36 processes needs to read in the entire global array of topography, a total of 188.28 GB of memory would be required for topography alone. This memory limitation of the centralized I/O approach would prevent a CONUS simulation from being executed using one process per node on Cheyenne's large memory nodes. As a result, parallelization has been limited to reading static inputs from binary files (as noted in the Appendix A), thereby preventing the need for global arrays of static input variables, excessive process communication, and memory allocation. Binary input files can be accessed concurrently by indexing the starting byte and length of bytes commensurate to a process's local domain. Therefore, each process only reads its own portion of the static input data.

Reading of meteorological forcing variables (wind speed, wind direction, relative humidity, temperature, and precipitation) can be performed in parallel with either binary or NetCDF files. Depending on the forcing dataset, the resolution of the meteorological variables typically ranges from 1 to 30 km and therefore is often much coarser than the static inputs for high-resolution simulations. For example, the resolution of NLDAS-2 meteorological forcing is approximately 11 km, while a high-resolution WRF model from NCAR is approximately 4 km. At each timestep, processes read in the forcing data from every station into a one-dimensional array, index the nearest station data, and interpolate the nearest station data to create forcing variables over the local domain. All processes perform the same operation and store common information (forcing data stored in the one-dimensional array). However, since the resolutions of the forcing datasets are significantly coarser than the model grid for high-resolution simulations, the dynamic forcing input array size remains comparable to other local arrays. While more efficient parallel file input schemes could improve performance, we decided to keep this logic to maintain consistency with the serial version of the code.

https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

3.4.2 Parallel Outputs

To eliminate the use of local to global mapping commonly used to output variables (Fig. 8a), each process writes its own output file (Fig. 8b). A postprocessing script is then used to concatenate files from each process into one file that represents the output for the global domain. Modern high-performance computing architectures have highly parallelized disc systems making file output using a distributed approach faster than the centralized approach. Therefore, file output in this manner reduces time and memory requirements.

4 Results

Parallel SnowModel experiments were evaluated on six domains in the United States (Fig. 1). All experiments were executed with a 100 m grid increment, a 3 h time step, and used a single-layer snowpack configuration. The validation experiments and scaling experiments were forced with NLDAS-2, while the CONUS simulation was forced with the higher-resolution WRF dataset. All experiments included the primary SnowModel modules (MicroMet, EnBal, SnowPack, and SnowTran-3D).

4.1 Parallel SnowModel Validation

A key requirement of Parallel SnowModel is for its simulation results to be identical to those from the serial algorithm. To assess the accuracy of the CAF implementation, two validation experiments were performed (Tuolumne and Colorado Headwaters). The Tuolumne domain is located in the Sierra Nevada Mountain Range in California and contains 311 and 185 grid cells in the x and y dimensions, respectively, while the Colorado Headwaters domain contains 3166 by 5167 grid cells in the x and y dimensions, respectively (Fig. 1). Both simulations were run at 100 m grid resolution using 3 h timesteps and forced with NLDAS-2 meteorological variables. The Tuolumne experiments were conducted from 1 September 2017 through 31 August 2018, for a total of 2920 timesteps. Due to the larger domain size and 12 h wall-clock limitation, the Colorado Headwaters validation experiments were simulated from 1 January 2018 through 1 February 2018, for a total of 256 timesteps.

The implementation of Parallel SnowModel was validated to assess the reproducibility of the results compared to the original serial model by varying the number of processes and therefore the size of the domain decomposition. We compared results from the original serial model to parallel simulations executed with 2, 4, 8, 16, 36, 52, 72, and 144 (Note: the maximum number of processes executed over the Tuolumne domain was 52 due to its domain size and decomposition). Comparisons were made on 17 output variables, including relevant snow variables like snow depth, SWE, snow density, and SWE-melt. A complete list of output variables is provided in Appendix B. We used the root mean square error (RMSE) metric to evaluate differences between results for each timestep from the parallel and sequential simulations (Eq. 3). All variables across all processes produced RMSE values of 10^{-6}, which is at the limit of machine precision, when compared to serial simulation results.

https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

$$RMSE(X,Y) = \sqrt{\frac{\sum(X_i - Y_i)^2}{N}}, \quad (3)$$

4.2 Parallel Performance

The performance of Parallel SnowModel was evaluated by comparing the execution time as a function of improvements to the algorithm (file I/O scheme and process communication) and execution time as a function of domain size and number of processes. Strong scaling is a common parallel performance metric implemented to understand the relationship between execution time and the number of processes used for a fixed domain size. The execution time used in the parallel performance assessments did not include the initialization of the algorithm (ReadParam, PreProcess, and array allocation), as to not weigh the initialization disproportionately, especially when running large domains over relatively short time periods. Speedup (S), a metric of strong scaling, is defined as the ratio of the serial execution time, $T(1)$, over the execution time using N cores, $T(N)$ (Eq. 4). Optimally, parallel algorithms will experience a doubling of speedup as the number of cores is doubled (e.g., ideal scaling).

$$S(N) = \frac{T(1)}{T(N)}, \quad (4)$$

Additionally, code profiling evaluates the execution time of individual submodules as a function of the number of processes. Together, code profiling and strong scaling can be used to understand locations of bottlenecks in the algorithm and how changes to the code affect performance. Figure 9 highlights the results of two significant changes made to the parallel algorithm, as shown through code profiling and speedup plots of three different stages of the code development (Mower et al., 2023). The simulations were executed on the Colorado Headwaters domain (Fig. 1) using 1, 2, 4, 8, 16, 36, 52, 72, and 144 processes, outputted one variable, and were forced with NLDAS-2 data for 16 timesteps from 23-24 March 2018. The first stage is a representation of the code when it used a centralized (Sect. 3.4) file I/O approach and is thus referred to as *Centralized* (Fig. 9). *Distributed High Sync* represents a version of the code with distributed file I/O and high or excessive process communication, while *Distributed Low Sync* represents a more recent version of the code where unnecessary parallel logic and communication had been removed (Fig. 9). As mentioned previously, synchronization calls (e.g., *sync all*) are necessary to accurately perform HX and for the parallel algorithm to achieve identical results as the serial algorithm (Sect. 4.1) but increase the overall execution time. Therefore, the major difference between the *Distributed High Sync* and the *Distributed Low Sync* is the optimization of process communication and wait times. The scaling results of the *Centralized* compared to the *Distributed Low Sync* version of the algorithm produced factors of 4 and 100 times speedup, respectively, when running with 144 processes. Code profiling plots of the *Centralized* version show the execution time of several submodules including ReadParam and Preproc (file input) and Output (file output) being constant as the number of processes increases. In other words, increasing the number of processes did not decrease the execution time within these submodules. Conversely, the execution time of ReadParam, Preproc, and Output all scale (decrease proportionately) with the number of processes in *Distributed High Sync*. SnowTran-3D displays an increase in execution time after approximately 36 processes

https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

(in both *Centralized* and *Distributed High Sync*), which is no longer observed in *Distributed Low Sync*. Therefore, the difference in speedup across these three stages is mainly attributed to bottlenecks occurring from file I/O schemes and excessive processor communication in SnowTran-3D. Ultimately, without these improvements, the CONUS domain could not be simulated using Parallel SnowModel.

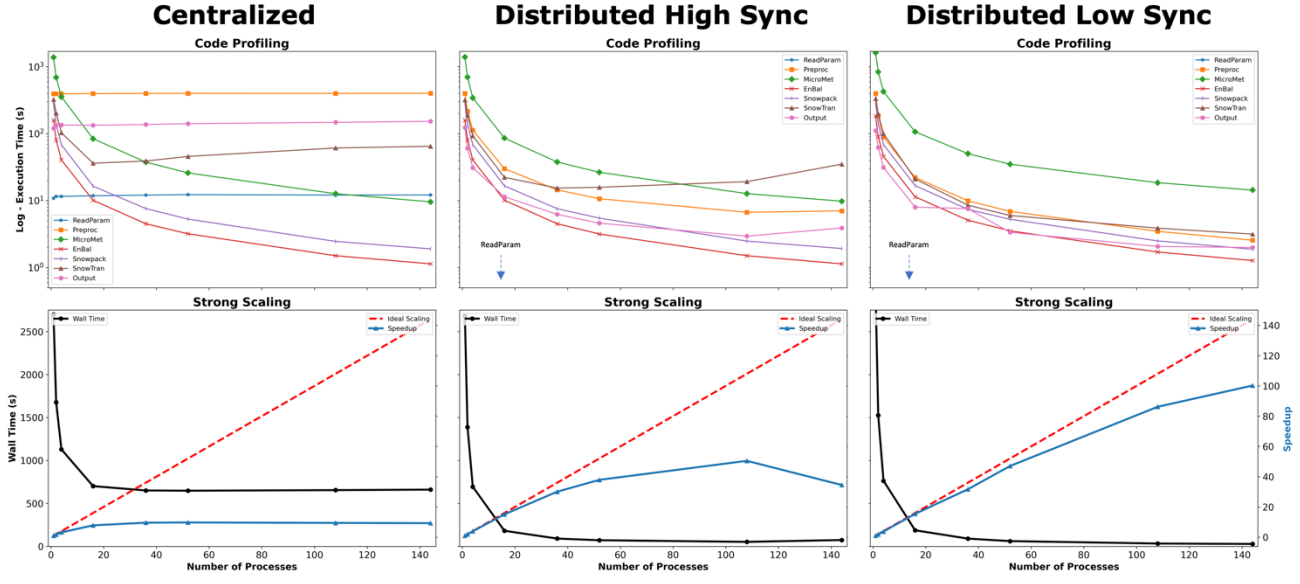

Figure 9: Code profiling (top row) and strong scaling (bottom row) results demonstrating the progression of Parallel SnowModel, which includes a version of the code with centralized file I/O (*Centralized*; first column), a version of the code with distributed I/O and high process communication (*Distributed High Sync*; second column), and a more recent version of the code which includes distributed file I/O and low process communication (*Distributed Low Sync*; third column). The arrow in the code profiling plots of *Distributed High Sync* and *Distributed Low Sync* indicates the ReadParam timing is below the y-axis at approximately 0.3 seconds and 0.003 seconds, respectively.

In addition to performing a scaling analysis across different versions of the code, we performed a scaling experiment across several domains using the current version of Parallel SnowModel (Mower et al., 2023). Six, 100 m resolution domains across the United States [Tuolumne, Colorado Headwaters, Idaho, PNW, Western U.S. (West), and CONUS] (Fig. 1) were simulated using different numbers of processes. The simulations were forced with NLDAS-2 data for 16 timesteps from 23-24 March 2018 and outputted one variable. While 16 timesteps is a short time period to perform scaling experiments, we wanted to compare timing metrics across different sized domains and were limited by memory and the 12 h wall-clock on Cheyenne. However, as mentioned previously, the initialization timing was removed in the speedup calculations. Additionally, over these selected dates, a significant amount of wind and frozen precipitation was observed over CONUS to activate some of the snow redistribution schemes in SnowTran-3D. Figure 10 shows the S as function of the number of processes for the local and state (Tuolumne, Colorado Headwaters, and Idaho Fig. 10a) and regional and continental sized domains (PNW, Western. U.S., and CONUS, Fig. 10b). For the regional and continental domains, where serial simulations could not be performed either due to wall-clock or memory limitations (as discussed in Sect. 2), the approximate speedup ($\hat{S}$)

https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

is estimated using the execution time, $T(\hat{P})$, of the simulation with the minimum number of processes ($\hat{P}$) by assuming perfect scaling from there to a single process (Eq. 5). For example, this experiment identified the $\hat{P}$ needed to run the PNW domain was 4 with a $T(\hat{1})$ of approximately 104 min (the total execution time including initialization was approximately 188 min). Therefore, the estimated $T(1)$, assuming ideal scaling, of running Idaho on one core would be 416 min. Near perfect scaling is evident up to 144 processes in most of the domains, so the assumption that scaling is linear below 52 processes ($T(\hat{1})$ for the CONUS domain) appears to be justified. While this approximation is an assumption, it is helpful to visualize the approximate $\hat{S}$ across the different domains on a similar scale.

$$\hat{S}(N) = \frac{T(\hat{P})}{T(N)} * \hat{P} \ , \tag{5}$$

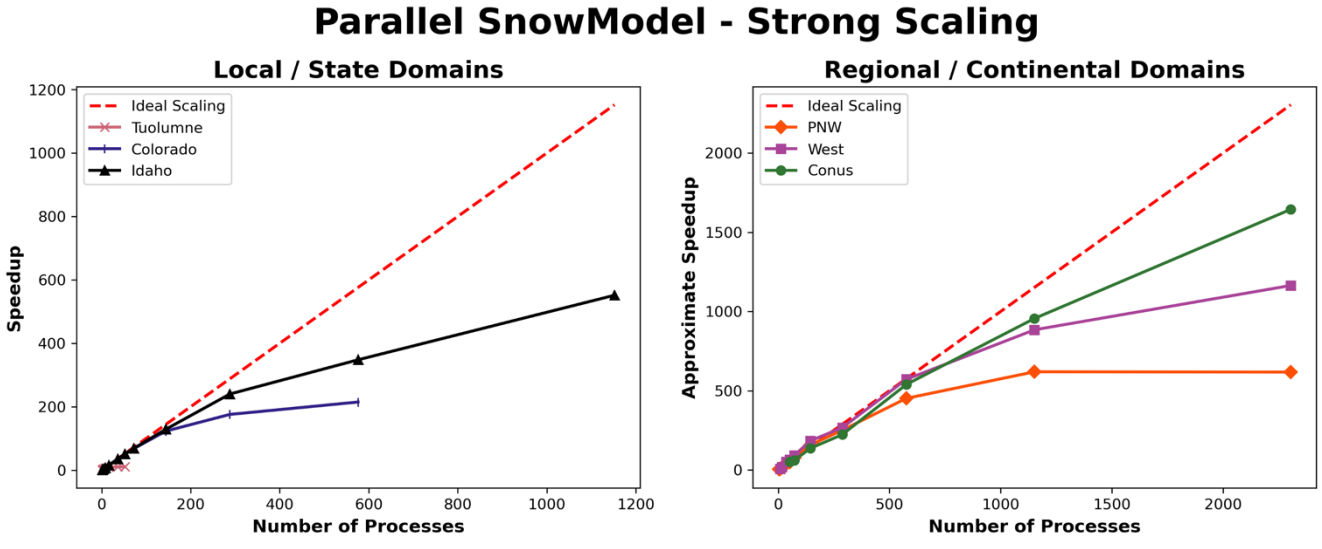

Figure 10: ***The left panel displays speedup (Eq. 4) for local and state sized simulations (Tuolumne, Colorado, and Idaho), while the right panel shows approximate speedup (Eq. 5) for the regional and continental sized domains (PNW, West, and CONUS).***

In strong scaling, the number of processes is increased while the problem size remains constant; therefore, it represents a reduced workload per process. Strong scaling analysis is useful for I/O and memory bound applications to identify a setup that results in a reasonable runtime and moderate resource costs (Fig. 10). The speedup obtained by increasing the number of processes above 288 for the Colorado Headwaters domain is marginal, while increasing the number of processes above 1152 results in an increase in runtime (decrease in speedup) for the PNW domain, due to excessive process communication. Local sized domains, e.g., Tuolumne, likely do not warrant the need for parallel resources because they have small serial runtimes and parallel efficiencies (*E;* Eq. 6), or approximate parallel efficiencies ($\hat{E}$; Eq. 7), which is the ratio of the speedup (Eq. 4) or approximate speedup (Eq. 5) to the number of processes (e.g., using 52 processes, Tuolumne had a *E* of 20%). However, state, regional, and continental domains stand to benefit more significantly from parallelization. The CONUS runtime

https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

decreased by a factor of 32 running on 2304 processes relative to 52 processes. Based on our approximate speedup assumption, we would expect a CONUS $\hat{S}$ of 1644 times on 2304 processes compared to one core, with an $\hat{E}$ of 71%. If the initialization portion of the algorithm (ReadParam, PreProcess, and array allocation) is included in the total execution time, the CONUS S increases to a factor of 47 running on 2304 processes relative to 52 processes. This actually results in superlinear scaling and is attributed to the initialization process, where the process count allows sections of data to be retained in the local cache, reducing the need for the process to interact with global memory as frequently. Additionally, if 17 variables, as opposed to one variable, are outputted, the CONUS $\hat{S}$ is reduced by 22%. Ultimately, the strong scaling analysis supports the effectiveness of running Parallel SnowModel at high-resolution over large domains.

$$E(N) = \frac{S}{N} * 100\% \ , \tag{6}$$

$$\hat{E}(N) = \frac{\hat{S}}{N} * 100\% \ , \tag{7}$$

4.3 CONUS Simulations

A primary goal of this work was to run Parallel SnowModel simulations for 21 years (2000 – 2021) over the Fig. 1 CONUS domain on a 100 m grid, while resolving the diurnal cycle in the model physics and creating a daily dataset of snow properties, including snow depth, SWE, SWE-melt, sublimation, and precipitation partitioning into rain and snow. Future work will involve analyzing results from these simulations. Ultimately, the domain contained 46,238 and 28,260 grid cells in the x and y dimensions, respectively. Simulations were performed on a 3 h time step and forced with the WRF dataset. All simulations were executed on Discover using 1800 processes with a total compute time of approximately 192,600 core hours, or approximately 5 wall-clock hours per year. Spatial results of SWE on 12 February 2011 over the CONUS domain and a sub-domain located in the Indian Peaks west of Boulder, Colorado are displayed in Fig. 11. The sub-domain highlights two grid cells located 200 m apart on a peak. The time series of SWE evolution for those grid cells (Fig. 11d and Fig. 11e) demonstrates the ability of Parallel SnowModel to capture fine-scale snow properties even when simulating continental domains. The upwind (western) grid cell is scoured by wind, and snow is transported to downwind (eastern) grid cells where a snow drift forms. The information and insight available in this high-resolution dataset will have important implications for many applications from hydrology, to wildlife and ecosystems, to weather and climate, and many more.

https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

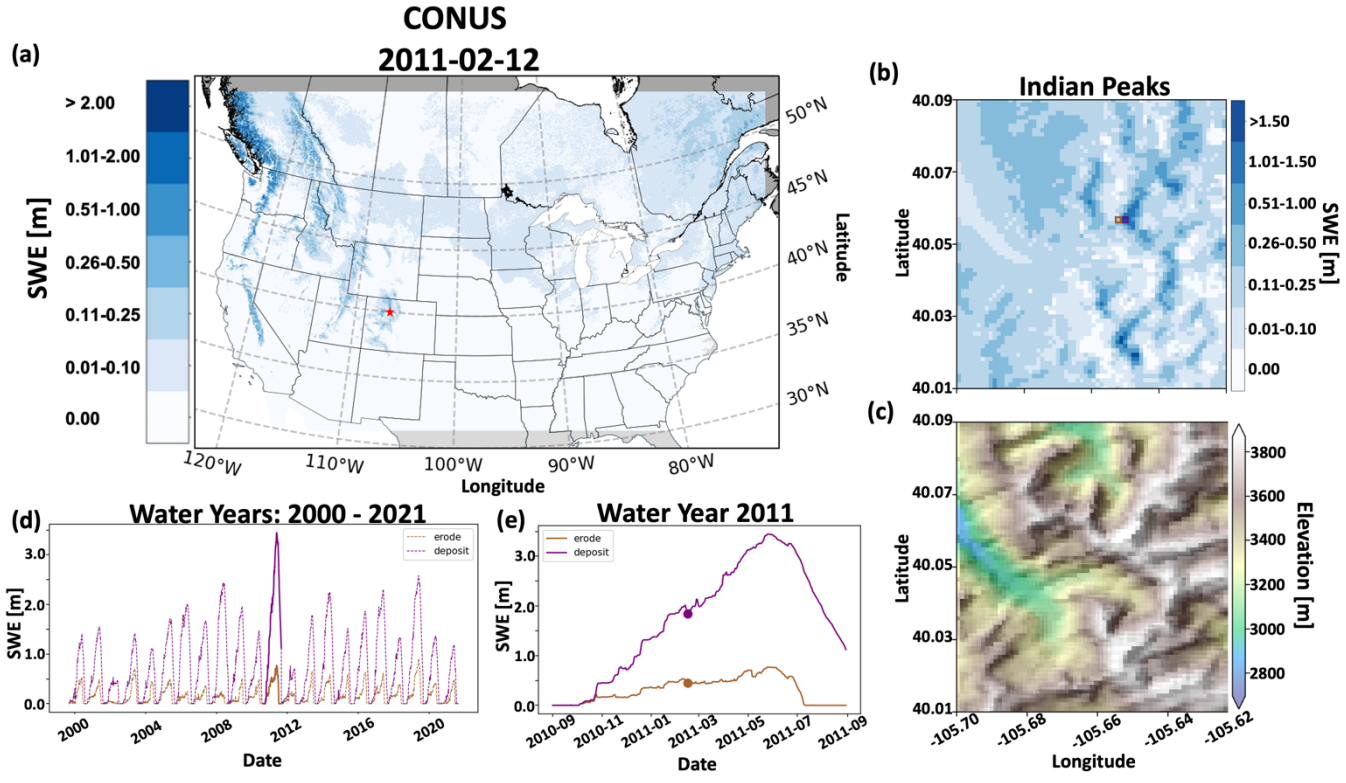

Figure 11: ***Simulation results of Parallel SnowModel over CONUS using WRF projection. Spatial patterns of SWE over the CONUS domain for 12 February 2011 (a), highlighting the SWE distribution (b) and topography with an applied hillshade (c) of a sub-domain near Apache Peak in the Indian Peaks west of Boulder, CO. Time series of SWE from 2000-2021 and over the 2011 water year for grid cells ("erode" and "deposit") identified in panel B are displayed in panels D and E, respectively. The "erode" and "deposit" grid cells highlight areas of similar elevation but significant differences in SWE evolution resulting from blowing-snow redistribution processes.***

5 Discussion

In this paper, we present a relatively simple approach that allows SnowModel to perform high-resolution simulations over regional to continental sized domains. The code within the core submodules (EnBal, MicroMet, SnowPack, and SnowTran-3D) and model configurations (single-layer snowpack, multi-layer snowpack, binary input files, etc.) were parallelized in this study. The parallelization subroutines of the program code have been modularized. This allows SnowModel to be compiled with Fortran compilers that do not support the Fortran 2008 standard, as well as modern compilers that support parallel CAF either internally or through libraries, such as OpenCoarrays (Fanfarillo et al., 2014). Additionally, it provides the structure for other parallelization logic (e.g., MPI) to be more easily added to the code base. The parallel module contains a simple approach to decomposing the computational domain in the *y* dimension into smaller rectangular sub-domains. These sub-domains are distributed across processes to perform asynchronous calculations. The parallelization module also contains logic for communicating information among processes using HX coarrays for the wind and solar radiation models,

https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

as well as for snow redistribution. These approaches can be adopted in other parallelization efforts where spatial derivatives are calculated or fluxes are transported across gridded domains.

Parallelizing numerical models often involves two-dimensional decomposition in both the *x* and *y* dimensions. While many benefits have been demonstrated by this approach, including improved load balancing (Dennis, 2007; Hamman et al., 2018), it comes with increased complication of the parallel algorithms, including the partitioning algorithm, file I/O, and process communication. The demonstrated speedup of Parallel SnowModel on high-performance computing architectures (Fig. 10), suggests that SnowModel scales effectively over regional to continental scales using the one-decomposition parallelization approach. The added benefits obtained from two-dimensional decomposition strategies might not outweigh the costs of development, testing, and minimizing changes to the code structure and logic for applications such as SnowModel. Ultimately, our simplified parallelization approach can be modeled by other geoscience schemes as a first step to enhance simulation size and resolution.

Simulation experiments were conducted using Parallel SnowModel to validate the parallel logic, interpret its performance across different versions of algorithm and across different sized domains, and demonstrate its ability to simulate continental domains at high-resolution. Most importantly, a comparison of output results from serial SnowModel and Parallel SnowModel validated the accuracy of the parallel algorithm and confirmed that the physical representations were not altered by the parallelization (Sect. 4.1). Code profiling and speedup analyses over the Colorado Headwaters domain helped identify bottlenecks in file I/O and processor communication in SnowTran-3D (Sect. 4.2). Corrections to the referred bottlenecks allowed Parallel SnowModel to scale up to regional and continental sized simulations. Parallel speedup analyses helped to identify the optimum number of processes and efficiency of the parallel algorithm for different domain sizes (Sect 4.2). Additionally, these experiments emphasize the relationships among speed, memory, and computing resources for Parallel SnowModel. A common laptop (~ 4 processes) has sufficient CPUs to run local sized domains within a reasonable amount of time, but likely does not have sufficient memory for state-sized simulations. Similarly, the minimum memory (1160 GB; Fig. 1) and processes (52; Fig. 10) required to run the CONUS domain, could be simulated on a large server (~ 128 processes) with one process per node. However, extrapolating from our scaling results on Cheyenne (Fig. 10), we estimate it would take over 10 days to run a CONUS simulation for one water year with this configuration. In contrast, it took approximately 5 hours for CONUS to run on the Discover supercomputer using 1800 processes (Sect. 4). Therefore, by the time it took the large server to complete a CONUS simulation for one water year, 48 water years could have been simulated on a supercomputer. Lastly, results from the CONUS simulation highlight the ability of Parallel SnowModel to run high-resolution continental simulations, while maintaining fine-scale snow processes that occur at a local level.

6 Conclusions

While several snow products exist, few capture the suite of snow properties along with the spatial and temporal extents and resolutions that can benefit a wide variety of applications. For example, current snow information products include the

https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

NASA daily SWE distributions globally for dry (non-melting) snow on a 25 km grid (Tedesco and Jeyaratnam, 2019), a NASA snow-cover product on a 500 m grid (Hall et al., 2006) that is often missing information due to clouds (approximately 50% of the time (Moody et al., 2005)), and the Snow Data Assimilation System (SNODAS) daily snow information provided by the National Oceanic and Atmospheric Administration (NOAA) and the National Weather Service (NWS) National Operational Hydrologic Remote Sensing Center (NOHRSC) on a 1 km grid (Center, 2004), which is itself model derived and has limited geographic coverage and snow properties. The Airborne Snow Observatory (ASO) provides the highest resolution data with direct measurements of snow depth on a 3 m grid, and derived values of SWE on a 50 m grid (Painter et al., 2016), but is flown on an aircraft and thus has limited spatio-temporal coverage. Furthermore, there are many fields of study that can benefit from 100 m resolution information of internally consistent snow variables, including wildlife and ecosystem, military, hydrology, weather and climate, cryosphere, recreation, remote sensing, engineering and civil works, and industrial applications. SnowModel can produce high-resolution outputs of snow depth, density, SWE, grain size, thermal resistance, snow strength, snow albedo, landscape albedo, meltwater production, snow-water runoff, blowing snow flux, visibility, peak winter SWE, snow-season length, snow onset date, snow-free date, and more, all produced by a physical model that maintains consistency among variables. The SnowModel system itself supports the assimilation of a wide variety of observations such that it can provide all of these variables while maintaining consistency with the limited *in situ* and remotely sensed measurements that are available. The new Parallel SnowModel described here permits the application of this modeling system to very large domains without sacrificing spatial resolution.

Appendix A

Some of the configuration combinations were not parallelized during this study for reasons including ongoing development in the serial code base and limitations to the parallelization approach. These include simulations involving tabler surfaces (Tabler, 1975), I/O using ASCII files, lagrangian seaice tracking, and data assimilation.

Appendix B

Validation SnowModel experiments were run in serial and in parallel over the Tuolumne and Colorado Headwaters domains (Sect. 4.1) using the RMSE statistic (Eq. 3). Important output variables from EnBal, MicroMet, SnowPack, and SnowTran-3D demonstrated similar, if not identical values, when compared to serial results for all timesteps during the simulations; RMSE values were within machine precision (~10^{-6}) regardless of the output variable, domain, or number of processes used. The validated output variables include albedo [%], precipitation $[m]$, emitted longwave radiation $[W * m^{-2}]$, incoming longwave radiation reaching the surface $[W * m^{-2}]$, incoming solar radiation reaching the surface $[W * m^{-2}]$, relative humidity [%], runoff from base of snowpack $[m * timestep]$, rain precipitation $[m]$, snow density $[kg * m^{-3}]$, snow-water equivalent melt $[m]$, snow depth $[m]$, snow precipitation $[m]$, static-surface sublimation $[m]$, snow-water equivalent $[m]$, air

https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

temperature [$°C$], wind direction [°], and wind speed [$m * s^{-1}$]. The Tuolumne domain could not be simulated with 72 processes, likely due to an insufficiently small local domain of 2-3 rows as a result of the domain decomposition. Ultimately, we feel confident that Parallel SnowModel is producing the same results as the original serial algorithm.

Code, data availability, and supplement

The Parallel SnowModel code and the data used in Sect. 4 is available through a public GitHub repository (Mower et al., 2023). The code base is limited to the parallelization changes to the serial version of the model. Furthermore, it does not contain preprocessing steps used to build simulation domains. For more information about the serial version of SnowModel, refer to Liston and Elder (2006b). The data includes figures and SnowModel output files that contain the necessary information to recreate the simulations. The gridded output variables themselves are not included due to storage limitations. Pending approval, we will submit our code to get a DOI.

Author contribution

EDG and GDL conceived the study. RM, EDG, GDL, and SR were integral in the code development. RM, EDG, and JL were involved in the design, execution, and interpretation of the experiments. All authors discussed the results and contributed to the final version of the draft.

Competing interests

The contact author has declared that none of the authors has any competing interests.

Disclaimer

Publisher's note: Copernicus Publications remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Financial support

The authors would like to acknowledge that this work has been performed under funding from NASA Earth Science Office (ESTO) Advanced Information Systems Technology (AIST) Program (grant no. 80NSSC20K0207) and support by the University of Washington's College of Engineering Fellowship.

https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

Acknowledgements

We acknowledge Alessandro Fanfarillo in his help during the early stages of the Parallel SnowModel code development. We are also grateful for the feedback from various team members involved in the AIST project, including Carrie Vuyovich, Kristi Arsenault, Melissa Wrzesien, Adele Reinking, and Barton Forman.

References

Beniston, M.: Climatic Change in Mountain Regions: A Review of Possible Impacts, Climatic Change, 59, 5-31, 10.1023/A:1024458411589, 2003.

Boelman, N. T., Liston, G. E., Gurarie, E., Meddens, A. J. H., Mahoney, P. J., Kirchner, P. B., Bohrer, G., Brinkman, T. J., Cosgrove, C. L., Eitel, J. U. H., Hebblewhite, M., Kimball, J. S., LaPoint, S., Nolin, A. W., Pedersen, S. H., Prugh, L. R., Reinking, A. K., and Vierling, L. A.: Integrating snow science and wildlife ecology in Arctic-boreal North America, Environmental Research Letters, 14, 010401, 10.1088/1748-9326/aaeec1, 2019.

Discover SCU Hardware: https://www.nccs.nasa.gov/systems/discover/scu-info, last

Center, N. O. H. R. S.: Snow data assimilation system (SNODAS) data products at NSIDC, 2004.

Clark, M. P. and Hay, L. E.: Use of Medium-Range Numerical Weather Prediction Model Output to Produce Forecasts of Streamflow, Journal of Hydrometeorology, 5, 15-32, 10.1175/1525-7541(2004)005<0015:Uomnwp>2.0.Co;2, 2004.

Coarfa, C., Dotsenko, Y., Mellor-Crummey, J., Cantonnet, F., El-Ghazawi, T., Mohanti, A., Yao, Y., and Chavarría-Miranda, D.: An evaluation of global address space languages: co-array fortran and unified parallel c, Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming, 36-47,

Dean B. Gesch, G. A. E., Michael J. Oimoen, Samantha Arundel: The National Elevation Dataset, 70201572, USGS Publication Warehouse2018.

Dennis, J. M.: Inverse space-filling curve partitioning of a global ocean model, 2007 IEEE International Parallel and Distributed Processing Symposium, 1-10,

Dozier, J., Bair, E. H., and Davis, R. E.: Estimating the spatial distribution of snow water equivalent in the world's mountains, WIREs Water, 3, 461-474, https://doi.org/10.1002/wat2.1140, 2016.

https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

570 Fanfarillo, A., Burnus, T., Cardellini, V., Filippone, S., Nagle, D., and Rouson, D.: OpenCoarrays: open-source transport
571 layers supporting coarray Fortran compilers, Proceedings of the 8th International Conference on Partitioned Global Address
572 Space Programming Models, 1-11,

573 Foster, J. L., Hall, D. K., Eylander, J. B., Riggs, G. A., Nghiem, S. V., Tedesco, M., Kim, E., Montesano, P. M., Kelly, R. E.
574 J., Casey, K. A., and Choudhury, B.: A blended global snow product using visible, passive microwave and scatterometer
575 satellite data, International Journal of Remote Sensing, 32, 1371-1395, 10.1080/01431160903548013, 2011.

576 Hall, D., Riggs, G., and Salomonson, V.: MODIS/Terra Snow Cover 5-Min L2 Swath 500m, Version, 5, 2011167.2011750,
577 2006.

578 Hamman, J. J., Nijssen, B., Bohn, T. J., Gergel, D. R., and Mao, Y.: The Variable Infiltration Capacity model version 5
579 (VIC-5): Infrastructure improvements for new applications and reproducibility, Geoscientific Model Development, 11, 3481-
580 3496, 2018.

581 Homer, C., Dewitz, J., Yang, L., Jin, S., Danielson, P., Xian, G., Coulston, J., Herold, N., Wickham, J., and Megown, K.:
582 Completion of the 2011 National Land Cover Database for the conterminous United States–representing a decade of land
583 cover change information, Photogrammetric Engineering & Remote Sensing, 81, 345-354, 2015.

584 Huss, M., Bookhagen, B., Huggel, C., Jacobsen, D., Bradley, R. S., Clague, J. J., Vuille, M., Buytaert, W., Cayan, D. R.,
585 Greenwood, G., Mark, B. G., Milner, A. M., Weingartner, R., and Winder, M.: Toward mountains without permanent snow
586 and ice, Earth's Future, 5, 418-435, https://doi.org/10.1002/2016EF000514, 2017.

587 Immerzeel, W. W., Lutz, A. F., Andrade, M., Bahl, A., Biemans, H., Bolch, T., Hyde, S., Brumby, S., Davies, B. J., Elmore,
588 A. C., Emmer, A., Feng, M., Fernández, A., Haritashya, U., Kargel, J. S., Koppes, M., Kraaijenbrink, P. D. A., Kulkarni, A.
589 V., Mayewski, P. A., Nepal, S., Pacheco, P., Painter, T. H., Pellicciotti, F., Rajaram, H., Rupper, S., Sinisalo, A., Shrestha,
590 A. B., Viviroli, D., Wada, Y., Xiao, C., Yao, T., and Baillie, J. E. M.: Importance and vulnerability of the world's water
591 towers, Nature, 577, 364-369, 10.1038/s41586-019-1822-y, 2020.

592 ISO/IEC: Fortran Standard 2008; Technical report, Geneva, Switzerland, 2010.

593 Jin, S., Homer, C., Yang, L., Danielson, P., Dewitz, J., Li, C., Zhu, Z., Xian, G., and Howard, D.: Overall methodology
594 design for the United States national land cover database 2016 products, Remote Sensing, 11, 2971, 2019.

595 Laboratory, C. a. I. S.: Cheyenne, 10.5065/D6RX99HX, 2019.

https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

Latifovic, R., Homer, C., Ressl, R., Pouliot, D., Hossain, S. N., Colditz, R. R., Olthof, I., Giri, C. P., and Victoria, A.: 20 North American Land-Change Monitoring System, Remote sensing of land use and land cover, 303, 2016.

Lettenmaier, D. P., Alsdorf, D., Dozier, J., Huffman, G. J., Pan, M., and Wood, E. F.: Inroads of remote sensing into hydrologic science during the WRR era, Water Resources Research, 51, 7309-7342, https://doi.org/10.1002/2015WR017616, 2015.

Liston, G., Reinking, A. K., and Boleman, N.: Daily SnowModel Outputs Covering the ABoVE Core Domain, 3-km Resolution, 1980-2020, 10.3334/ORNLDAAC/2105, 2022.

Liston, G. E.: Local advection of momentum, heat, and moisture during the melt of patchy snow covers, Journal of Applied Meteorology and Climatology, 34, 1705-1715, 1995.

Liston, G. E.: Representing Subgrid Snow Cover Heterogeneities in Regional and Global Models, Journal of Climate, 17, 1381-1397, 10.1175/1520-0442(2004)017<1381:Rsschi>2.0.Co;2, 2004.

Liston, G. E. and Elder, K.: A meteorological distribution system for high-resolution terrestrial modeling (MicroMet), Journal of Hydrometeorology, 7, 217-234, 2006a.

Liston, G. E. and Elder, K.: A distributed snow-evolution modeling system (SnowModel), Journal of Hydrometeorology, 7, 1259-1276, 2006b.

Liston, G. E. and Hall, D. K.: An energy-balance model of lake-ice evolution, Journal of Glaciology, 41, 373-382, 1995.

Liston, G. E. and Hiemstra, C. A.: The changing cryosphere: Pan-Arctic snow trends (1979–2009), Journal of Climate, 24, 5691-5712, 2011.

Liston, G. E. and Mernild, S. H.: Greenland freshwater runoff. Part I: A runoff routing model for glaciated and nonglaciated landscapes (HydroFlow), Journal of Climate, 25, 5997-6014, 2012.

Liston, G. E. and Sturm, M.: A snow-transport model for complex terrain, Journal of Glaciology, 44, 498 - 516, 1998.

Liston, G. E. and Sturm, M.: Global Seasonal-Snow Classification, Version 1 [dataset], 2021.

Liston, G. E., Perham, C. J., Shideler, R. T., and Cheuvront, A. N.: Modeling snowdrift habitat for polar bear dens, Ecological Modelling, 320, 114-134, https://doi.org/10.1016/j.ecolmodel.2015.09.010, 2016.

https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

Liston, G. E., Winther, J.-G., Bruland, O., Elvehøy, H., and Sand, K.: Below-surface ice melt on the coastal Antarctic ice sheet, Journal of Glaciology, 45, 273-285, 1999.

Liston, G. E., Haehnel, R. B., Sturm, M., Hiemstra, C. A., Berezovskaya, S., and Tabler, R. D.: Simulating complex snow distributions in windy environments using SnowTran-3D, Journal of Glaciology, 53, 241-256, 2007.

Liston, G. E., Itkin, P., Stroeve, J., Tschudi, M., Stewart, J. S., Pedersen, S. H., Reinking, A. K., and Elder, K.: A Lagrangian snow-evolution system for sea-ice applications (SnowModel-LG): Part I—Model description, Journal of Geophysical Research: Oceans, 125, e2019JC015913, 2020.

Mahoney, P. J., Liston, G. E., LaPoint, S., Gurarie, E., Mangipane, B., Wells, A. G., Brinkman, T. J., Eitel, J. U., Hebblewhite, M., and Nolin, A. W.: Navigating snowscapes: scale-dependent responses of mountain sheep to snowpack properties, Ecological Applications, 28, 1715-1729, 2018.

Miller, P., Robson, M., El-Masri, B., Barman, R., Zheng, G., Jain, A., and Kalé, L.: Scaling the isam land surface model through parallelization of inter-component data transfer, 2014 43rd International Conference on Parallel Processing, 422-431,

Mitchell, K. E.: The multi-institution North American Land Data Assimilation System (NLDAS): Utilizing multiple GCIP products and partners in a continental distributed hydrological modeling system, J. Geophys. Res., 109, D07S90, 2004.

Moody, E. G., King, M. D., Platnick, S., Schaaf, C. B., and Gao, F.: Spatially complete global spectral surface albedos: Value-added datasets derived from Terra MODIS land products, IEEE Transactions on Geoscience and Remote Sensing, 43, 144-158, 2005.

Morin, S., Horton, S., Techel, F., Bavay, M., Coléou, C., Fierz, C., Gobiet, A., Hagenmuller, P., Lafaysse, M., Ližar, M., Mitterer, C., Monti, F., Müller, K., Olefs, M., Snook, J. S., van Herwijnen, A., and Vionnet, V.: Application of physical snowpack models in support of operational avalanche hazard forecasting: A status report on current implementations and prospects for the future, Cold Regions Science and Technology, 170, 102910, https://doi.org/10.1016/j.coldregions.2019.102910, 2020.

Mower, R., Gutmann, E. D., and Liston, G. E.: Parallel-SnowModel 1.0 [code], https://github.com/NCAR/Parallel-SnowModel-1.0, 2023.

Mudryk, L. R., Derksen, C., Kushner, P. J., and Brown, R.: Characterization of Northern Hemisphere Snow Water Equivalent Datasets, 1981–2010, Journal of Climate, 28, 8037-8051, 10.1175/jcli-d-15-0229.1, 2015.

https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

Nolin, A. W.: Recent advances in remote sensing of seasonal snow, Journal of Glaciology, 56, 1141-1150, 10.3189/002214311796406077, 2010.

Numrich, R. W. and Reid, J.: Co-Array Fortran for parallel programming, ACM Sigplan Fortran Forum, 1-31,

Numrich, R. W., Steidel, J. L., Johnson, B. H., Dinechin, B. D. d., Elsesser, G., Fischer, G., and MacDonald, T.: Definition of the F−− Extension to Fortran 90, International Workshop on Languages and Compilers for Parallel Computing, 292-306,

Painter, T. H., Berisford, D. F., Boardman, J. W., Bormann, K. J., Deems, J. S., Gehrke, F., Hedrick, A., Joyce, M., Laidlaw, R., and Marks, D.: The Airborne Snow Observatory: Fusion of scanning lidar, imaging spectrometer, and physically-based modeling for mapping snow water equivalent and snow albedo, Remote Sensing of Environment, 184, 139-152, 2016.

Parhami, B.: SIMD machines: do they have a significant future?, ACM SIGARCH Computer Architecture News, 23, 19-22, 1995.

Perezhogin, P., Chernov, I., and Iakovlev, N.: Advanced parallel implementation of the coupled ocean–ice model FEMAO (version 2.0) with load balancing, Geoscientific Model Development, 14, 843-857, 2021.

Pflug, J. M. and Lundquist, J. D.: Inferring Distributed Snow Depth by Leveraging Snow Pattern Repeatability: Investigation Using 47 Lidar Observations in the Tuolumne Watershed, Sierra Nevada, California, Water Resources Research, 56, e2020WR027243, https://doi.org/10.1029/2020WR027243, 2020.

Rasmussen, R. M., Liu, C., Ikeda, K., Chen, F., Kim, J.-H., Schneider, T., Gochis, D., Dugger, A., and Viger, R.: Four-kilometer long-term regional hydroclimate reanalysis over the conterminous United States (CONUS), 1979-2020, Research Data Archive at the National Center for Atmospheric Research, Computational and Information Systems Laboratory [dataset], 10.5065/ZYY0-Y036, 2023.

Renwick, J.: MOUNTerrain: GEWEX mountainous terrain precipitation project, GEWEX news, 24, 5-6, 2014.

Richter, B., Schweizer, J., Rotach, M. W., and van Herwijnen, A.: Modeling spatially distributed snow instability at a regional scale using Alpine3D, Journal of Glaciology, 67, 1147-1162, 10.1017/jog.2021.61, 2021.

Rouson, D., Gutmann, E. D., Fanfarillo, A., and Friesen, B.: Performance portability of an intermediate-complexity atmospheric research model in coarray Fortran, Proceedings of the Second Annual PGAS Applications Workshop, 1-4,

Sharma, V., Swayne, D., Lam, D., MacKay, M., Rouse, W., and Schertzer, W.: Functional Parallelization of a Land Surface Model in Regional Climate Modeling, 2004.

https://doi.org/10.5194/egusphere-2023-1612
Preprint. Discussion started: 24 July 2023

673 Skofronick-Jackson, G. M., Johnson, B. T., and Munchak, S. J.: Detection Thresholds of Falling Snow From Satellite-Borne
674 Active and Passive Sensors, IEEE Transactions on Geoscience and Remote Sensing, 51, 4177-4189,
675 10.1109/TGRS.2012.2227763, 2013.

676 Sturm, M. and Liston, G. E.: Revisiting the global seasonal snow classification: An updated dataset for earth system
677 applications, Journal of Hydrometeorology, 22, 2917-2938, 2021.

678 Tabler, R. D.: Estimating the transport and evaporation of blowing snow, Great Plains Agric Counc Publ, 1975.

679 Takala, M., Luojus, K., Pulliainen, J., Derksen, C., Lemmetyinen, J., Kärnä, J.-P., Koskinen, J., and Bojkov, B.: Estimating
680 northern hemisphere snow water equivalent for climate research through assimilation of space-borne radiometer data and
681 ground-based measurements, Remote Sensing of Environment, 115, 3517-3529, https://doi.org/10.1016/j.rse.2011.08.014,
682 2011.

683 Tedesco, M. and Jeyaratnam, J.: AMSR-E/AMSR2 Unified L3 Global Daily 25 km EASE-Grid Snow Water Equivalent,
684 Version 1, Boulder, Colorado USA, NASA National Snow and Ice Data Center Distributed Active Archive Center, 2019.

685 Vuyovich, C. M., Jacobs, J. M., and Daly, S. F.: Comparison of passive microwave and modeled estimates of total watershed
686 SWE in the continental United States, Water Resources Research, 50, 9088-9102, https://doi.org/10.1002/2013WR014734,
687 2014.

688 Wrzesien, M. L., Durand, M. T., Pavelsky, T. M., Kapnick, S. B., Zhang, Y., Guo, J., and Shum, C. K.: A New Estimate of
689 North American Mountain Snow Accumulation From Regional Climate Model Simulations, Geophysical Research Letters,
690 45, 1423-1432, https://doi.org/10.1002/2017GL076664, 2018.

691 Xia, Y.: Continental-scale water and energy flux analysis and validation for North American Land Data Assimilation System
692 project phase 2 (NLDAS-2): 1. Intercomparison and application of model products, J. Geophys. Res., 117, D03109, 2012a.

693 Xia, Y.: Continental-scale water and energy flux analysis and validation for North American Land Data Assimilation System
694 project phase 2 (NLDAS-2): 2. Validation of model-simulated streamflow, J. Geophys. Res., 117, D03110, 2012b.
695