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

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

[1]The NSF 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 100s of meters) simulations over large, regional to continental scale, domains. In the parallel algorithm, the model domain was split into smaller rectangular sub-domains that are distributed over multiple processor cores using one-dimensional decomposition. All the memory allocations from the original code were reduced to the size of the local sub-domains, allowing each core to perform fewer computations and requiring less memory for each process. Most 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) for a year. The CONUS scaling experiment had approximately 70% parallel efficiency; runtime decreased by a factor of 1.9 running on 1800 cores relative to 648 cores (the minimum number of cores that could be used to run such a large domain because 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).

Deleted: Glen E. Liston[3],
Formatted: Not Superscript/ Subscript
Deleted: 100's
Deleted: is
Deleted: of
Deleted: have been
Deleted: A majority
Deleted:).
Deleted: 71
Deleted: 32
Deleted: 2304
Deleted: 52
Deleted: as a result

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 sensors, 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).

Deleted: observations

Many physical snow models have been developed either in stand-alone algorithms or larger LSMs with varying degrees of complexity based on their application. The more advanced algorithms attempt to accurately model snow properties at higher resolution especially in regions where snow interacts with topography, vegetation, and/or wind. Wind-induced snow transport is one such complexity of snow that represents an important interaction between the cryosphere and atmosphere. It occurs in regions permanently or temporarily covered by snow and greatly influences snow heterogeneity, sublimation, avalanches, and melt timing. Models that have incorporated wind-induced physics generally require components to both develop the snow mass balance and incorporate atmospheric inputs of the wind field. However, there often exists a trade-off between the accuracy of simulating wind-induced snow transport and the computational requirements for downscaling and developing the wind fields over the gridded domain (Reynolds et al., 2021; Vionnet et al., 2014). Therefore, simplifying assumptions of uniform wind direction has been applied in models like Distributed Blowing Snow Model (DBSM) (Essery et al., 1999; Fang and Pomeroy, 2009). More advanced models have utilized advection-diffusion equations, like Alpine3D (Lehning et al., 2006) or spatial distributed formulations like SnowTran-3D (Liston and Sturm, 1998). Finite volume methods for more efficiently discretizing wind fields have been applied to models such as DBSM (Marsh et al., 2020). The most complex models consider nonsteady turbulence which utilize three-dimensional wind fields from atmospheric models to simulate blowing snow transport and sublimation; for example, SURFEX in Meso-NH/Crocus (Vionnet et al., 2014; Vionnet et al., 2017), wind fields from the atmospheric model ARPS (Xue et al., 2000) being incorporated into Alpine3D (Mott and Lehning, 2010; Mott et al., 2010; Lehning et al., 2008), and SnowDrift3D (Prokop and Schneiderbauer, 2011).

Incorporating wind-induced physics into snow models is computationally expensive; thus, parallelizing the serial algorithms would likely be beneficial to many models.

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, 2006a; 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). Physically derived snow algorithms, as used in SnowModel, that model the energy balance, multilayer snow physics, and lateral snow transport are 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 can run one or more processes in parallel.

Deleted: (Liston and Elder, 2006b; Liston et al., 2020)

Deleted: Models like

Deleted: can be

Deleted: -

Deleted: is able to

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 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, high-resolution (100 m) distributed snow datasets over CONUS,

Deleted: and

and an analysis of the performance of the parallel algorithm; 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 techniques.

In Sect. 2, we provide background information on SnowModel, parallelization using CAF, data and domains used in this study, and a motivation for this work. In Sect. 3, we explain our parallelization approach using CAF and introduce the simulation experiments used to demonstrate the In Sect. performance of Parallel SnowModel through strong scaling metrics and CONUS simulations 4, we provide results of the simulation experiments introduced in Sect. 3. Lastly, we end with a discussion in Sect. 5 and a conclusion in Sect. 6.

2 Background

2.1 SnowModel

SnowModel is a spatially distributed snow-evolution modeling system designed to model snow states (e.g., snow depth, SWE, snow melt, snow density) and fluxes over different landscapes and climates (Liston and Elder, 2006a). The most complete and up-to-date description of SnowModel can be found in the Appendices of Liston et al. (2020). While many snow modelling systems exist, SnowModel will benefit from parallelization because of its ability to simulate snow processes on a high-resolution grid through downscaling meteorological inputs and modelling snow redistribution. SnowModel is designed to simulate domains on a structured grid with spatial resolutions ranging from 1 to 200 m (although it can simulate coarser resolutions, as well) and temporal resolutions ranging from 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, core interacting submodules: MicroMet defines the meteorological forcing conditions (Liston and Elder, 2006b), EnBal describes surface and energy exchanges (Liston, 1995; Liston et al., 1999), SnowPack-ML 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). Additional simulation features include SnowDunes (Liston et al., 2018) and SnowAssim (Liston and Hiemstra, 2008), which model sea-ice applications and data assimilation techniques, respectively. Figure 1 shows a schematic of the core SnowModel toolkit. Additionally, the initialization submodules that read in the model parameters, distribute inputs across the modeled grid, allocate arrays, etc., include PreProcess and ReadParam. Outputting arrays is contained within the Outputs submodule. 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)].

Deleted: (HX)
Deleted: describe
Deleted: provide
Deleted: its parallelization.
Moved (insertion) [1]
Deleted: the module developed that partitions
Deleted: two-dimensional domain in the *y* dimension and organizes the non-trivial communication necessary to produce accurate results. In Sect
Deleted: 4, we validate results from
Deleted: 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
Moved up [1]: In Sect.
Deleted: over several basins throughout the United States,
Deleted: present Parallel SnowModel results over
Deleted: .
Formatted: Heading 2
Deleted: properties
Deleted: (Liston and Elder, 2006b).
Deleted: modeling
Deleted: is standing to
Deleted: across larger domains
Formatted: Font: (Default) Times New Roman
Deleted: input
Deleted: data (e.g., wind, radiation, and precipitation)
Deleted: model fine-scale
Deleted: properties and
Deleted: processes
Formatted: Font: (Default) Times New Roman
Formatted: Font: (Default) Times New Roman
Formatted: Font: (Default) Times New Roman
Formatted: Font: (Default) Times New Roman
Formatted: Font: (Default) Times New Roman
Deleted: has the ability to
Deleted: (Liston and Elder, 2006a)
Deleted: .
Deleted: .

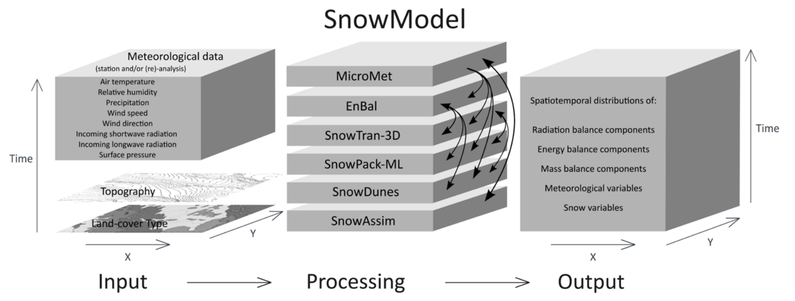


Figure 1: Schematic modified by Pederson et al. (2015) providing an example of possible inputs, core submodules, and outputs of SnowModel.

2.2 Coarray Fortran

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 like 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.

Moved (insertion) [2]

Moved (insertion) [3]

2.3 Model Domains, Data, and Computing Resources

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 introduced in Sect. 3: USGS National Elevation Dataset (NED) for topography (Gesch et al., 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-

Deleted: conducted

Deleted: 4

Deleted: (Dean B. Gesch, 2018)

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. Simulation experiments were conducted over six domains (Tuolumne, CO Headwaters, Idaho, PNW, Western US, and CONUS) throughout the United States at 100 m grid resolution. The spatial location, domain dimensions (e.g., number of grids in the *x* and *y* dimensions), and memory requirements, derived from the peak_memusage package (https://github.com/NCAR/peak_memusage), for the simulation experiments are highlighted in Figure 2.

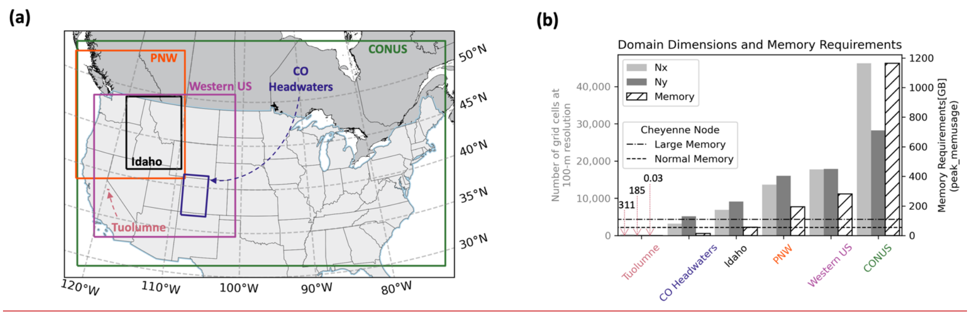


Figure 2: (a) Spatial location of simulated domains on WRF's lambert conformal projection (Rasmussen et al., 2023) and (b) corresponding grid dimensions (Nx – number of grids in x dimension; Ny – number of grids in y dimension) and memory obtained from peak_memusage package required for single-layer SnowModel simulation experiments. For reference, the dashed lines represent the normal and large memory thresholds (55 and 109 GB) for Cheyenne's SGI ICE XA cluster.

2.4 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 SnowModel (Sect. 4), serial simulations were attempted over six domains throughout the United States at 100 m grid resolution (Figure 2) for the 2018 water year (1 September 2017 to 1 September 2018). Only the Tuolumne domain could be simulated in serial based on the memory (109 GB for a large memory node) and time (12 h wall-clock limit) constraints on Cheyenne. The CO Headwaters and Idaho domains could not be simulated in serial due to time constraints, while the three largest domains (Pacific Northwest (PNW), Western U.S. and CONUS) could not be executed in serial due to both exceedances of the 12 h wall-clock limit and memory availability. Furthermore, we estimate that using a

Deleted: (Carriere, 2023).

Moved (insertion) [4]

Deleted: 2.1

Deleted: parallel algorithm

Deleted: . The spatial location,

Deleted: dimensions (e.g., number of grids

Deleted: the *x*

Deleted: *y* dimensions),

Deleted: memory requirements, derived from the peak_memusage package (https://github.com/NCAR/peak_memusage), for the simulation experiments are highlighted in

Deleted: Fig. 1. The simulations were executed on Cheyenne for 16 timesteps on 23-24 March 2018 using NLDAS-2 forcing. The

Deleted: using Cheyenne's normal or large memory (55 GB and 109 GB, respectively) compute nodes due to both exceedances of the 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. 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.

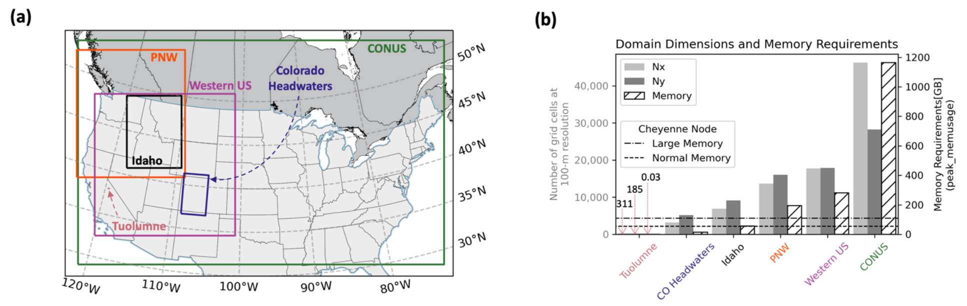


Figure 3:3 Methods

In parallelizing SnowModel and distributing computations and memory over multiple processes, we demonstrate its ability to efficiently run regional to continental sized simulations. 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 and the simulation experiments used to assess the performance of the parallel algorithm.

3.1 Parallel Implementation

Changes to the SnowModel logic were made through the parallelization process and included the partitioning algorithm, non-trivial communication via halo-exchange, and file input and output (I/O) schemes.

3.1.1 Partitioning Algorithm

The partitioning strategy identifies how the workload gets distributed amongst processes in a parallel algorithm. The multidimensional arrays of SnowModel are stored in row-major order, meaning 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. Therefore, both the data structures and physical processes involved in SnowModel justify

Deleted: (Fig. 1) at a 100 m grid increment and a 3 h time step.

Moved up [4]: ***Spatial location of simulated domains on WRF's lambert conformal projection (Rasmussen et al., 2023) (a)***

Deleted: ***(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***

Deleted: ***represent the normal and large memory thresholds (55 and 109 GB) for Cheyenne's SGI ICE XA cluster.***¶

Deleted: **Parallel Approach**

Deleted: hope

Deleted: be able to

Deleted: efficiently

Deleted: , including the partitioning strategy, algorithms involving non-trivial processor communication via halo exchange, and

Deleted: file input and output (I/O).

Deleted: **Fortran Coarrays**

Moved up [2]: 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.

Moved up [3]: 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. ¶

Deleted: CAF is similar to

Deleted: 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 infor ... [1]

Deleted: Algorithm 1 (Fig. 2) demonstrates a CAF program executed with three processes. In this program memory from ... [2]

Deleted: 2

Formatted: Heading 3

Deleted: Both the data structures and physical processes involved in SnowModel justify a one-dimensional decomposition strat ... [3]

Deleted: that

Formatted: Font: Courier New, Not Italic a one-dimensional decomposition strategy in the y dimension, where 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 the total number of processes (`N`), `l_ny = N_y / N`. If integer division is not possible, the remaining rows are distributed evenly amongst the processes starting at the bottom of the computational domain. Figure 4 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.

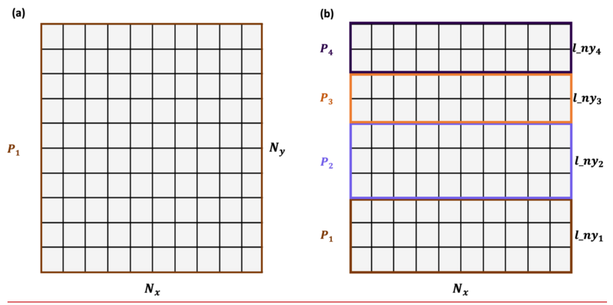


Figure 4: ***Example 10 x 10 global domain and partitioning for (a) serial simulation and (b) parallel simulation using four processes.***

3.1.2 Non-trivial Parallelization

Each process has sufficient information to correctly execute most of the physical computations within SnowModel. However, there are certain subroutines where grid computations require information from neighboring grid cells (e.g., data dependencies) and therefore information outside of the local domain of a process. 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 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 halo-exchange where coarrays are used in remote calls. Halo-exchange using CAF involves copying boundary data into coarrays on neighboring processes and using information from the coarrays to complete computations (Figure 5). Although the entire local array could be declared a coarray and accessed by remote processes more directly, some CAF implementations, (e.g. Cray) impose additional constraints upon coarray memory allocations that can be problematic for such large allocations.

Deleted: The partitioning algorithm decomposes the domai ... [4]

Deleted: N_y

Deleted: N_x

Deleted: l_ny

Deleted: N_y

Deleted: N_p, then $l\_ny_p = \frac{N_y}{N_p}$.

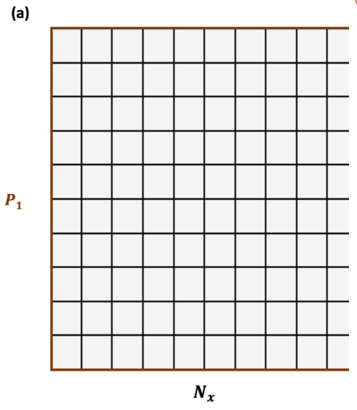
Formatted: Font: (Default) Times New Roman, Not Italic

Formatted: Font: Courier New

Deleted: then

Deleted: Figure 3

Formatted ... [5]



$partition\_ny = [N_y]$
$prefix\_sum = [1]$

Deleted:

Formatted: Font: Italic, English (US)

Formatted: Caption, Centered, Don't keep with next

Deleted: **Figure 3:** ***Example 10 x 10 global domain and partitioning for serial simulation (a), and parallel simulation using four processes (b).***¶
3.3

Formatted: Heading 3

Deleted: a majority of the physical computations within ... [6]

Deleted: (Fig. 4).

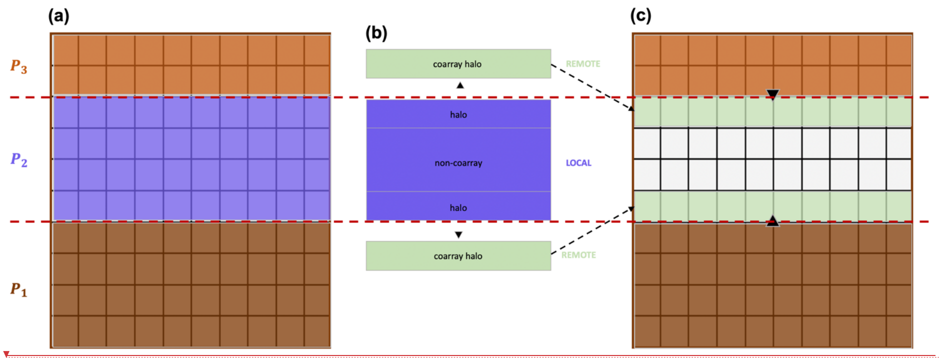
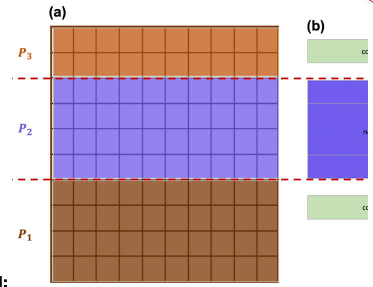


Figure 5: ***Schematic showing halo-exchange using coarrays. The steps include: (a) initial gridded representation of local arrays for three processes, (b) P_2 copying boundary data into coarrays for remote access, (c) neighboring processes (P_1 and P_3) stitching coarray to local domains.***

3.1.2.1 Topography – Wind and Solar Radiation Models

The wind and solar radiation models in MicroMet require information about surrounding surface topography (Liston and Elder, 2006b). 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 surface 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. 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. Note that the number of row(s) (`inc`) is determined by a predefined parameter that represents the wavelength of topographic features within a domain. Future work should permit this parameter to vary spatially to account for changes in the length scale across the domain. For example, all grid cells along the top row of `P1` 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 `P2` to make the calculation (Figure 6a). Halo-exchange is performed to distribute row(s) of data to each process that is missing that information in their local domains (Figure 6b). Processes whose local domains are positioned in the bottom or top of the global domain will only perform one halo-exchange with their interior neighbor, while interior processes will perform two halo-exchanges. By combining and appropriately indexing information from the process local array and received coarrays of topographic elevation, an accurate curvature calculation can be performed using this parallel approach (Figure 6c).



Deleted:

Formatted: Centered

Deleted: *HX*

Deleted: *(a),*

Deleted: *(b),*

Formatted: Font: Courier New

Formatted: Font: Courier New

Formatted: Font: Courier New

Deleted: *(c).*

Deleted: 3

Formatted: Heading 4

Deleted: .

Deleted: ,

Deleted: ($Z_w, Z_E, Z_S, Z_N, Z_{SW}, Z_{NE}, Z_{NW}, Z_{SE}$ [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).¶ ... [7]

Deleted: P_1

Deleted: *topo*

Deleted: P_2

Deleted: (Algorithm 2, Fig. 5a). HX

Deleted: (*inc*) of *topo*

Deleted: (Fig. 5b).

Deleted: HX

Deleted: HXs.

Deleted: process's

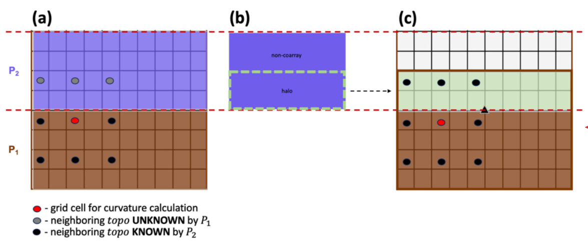
Deleted: (*coarray_n* and *coarray_s*)

Deleted: (Fig. 5c). Lastly, *sync all* (Algorithm 2) is an intri ... [8]

```
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
```



(a)
(b)
(c)
P_2
P_1
non-coarray
halo
● - grid cell for curvature calculation
● - neighboring *topo* **UNKNOWN** by P_1
● - neighboring *topo* **KNOWN** by P_2

Figure 6: Schematic for halo-exchange used in the curvature calculation by P_1, where inc = 2. (a) Prior to halo-exchange, P_1 contains insufficient information to perform the curvature calculation, (b) grid cells (halo) within the local domain of P_2 are (c) transferred to P_1 via coarrays. At this point, P_1 has sufficient information to make the curvature calculation.

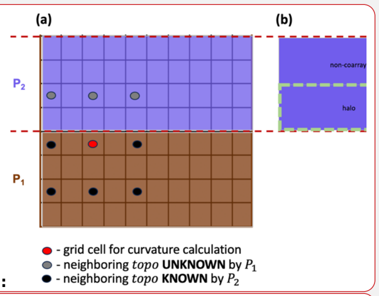
3.1.2.2 Snow Redistribution

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 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 7 shows a simplified schematic for the saltation flux from a southerly wind. In the serial algorithm (Figure 7a), SnowModel initializes the saltation flux based on the wind speed at that time step (`initial flux`). To calculate the final saltation flux (`updated flux`), SnowModel steps through regions of continuous wind direction (delineated by the indices: `jstart` and `jend`), updates the change in saltation fluxes from upwind grid cells and the change in saltation flux from the given wind direction, and makes adjustments to these fluxes based on the soft snow availability above the vegetation height (Liston and Elder, 2006a). Similar logic is used for the parallel implementation of the saltation and suspension fluxes with an additional iteration (`salt_iter`) that updates the boundary condition for each process via halo-exchange. This allows the fluxes to be communicated from the local domain of

Deleted:



(a)
(b)
P_2
P_1
non-coarray
halo
● - grid cell for curvature calculation
● - neighboring *topo* **UNKNOWN** by P_1
● - neighboring *topo* **KNOWN** by P_2

Formatted: Centered

Deleted: HX of *coarray_s*

Deleted: P_1,

Deleted: *inc*

Deleted: HX, P_1

Formatted: Font: Times New Roman, Subscript

Deleted: (a),…grid cells (halo) within the local domain ... [9]

Deleted: P_1

Deleted: (coarray_s, Alg. 2) (c). At this point, $P_1 P_1$ ... [10]

Deleted: 3

Formatted: Heading 4

Deleted: Figure 6 shows a simplified serial algorithm and ... [11]

Deleted: To start, SnowModel initializes the maximum ... [12]

Deleted:) of the same wind direction, updates the change ... [14]

Formatted ... [13]

Deleted: _

Deleted: to update the boundary condition for each proces ... [15]

Formatted: Font: Courier New

Formatted: Font: Courier New

one process to another. To minimize the number of iterations, `salt_iter` was provided a maximum bound that is equivalent to snow being transported 15 km via saltation or suspension. 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 schematic of the parallel calculation of the change in saltation due to southerly winds is illustrated in Figure 7b. The `bc_halo_exchange` represents a halo-exchange of grid cells from upwind processes, allowing the saltation flux to be transported from one process local domain to the next.

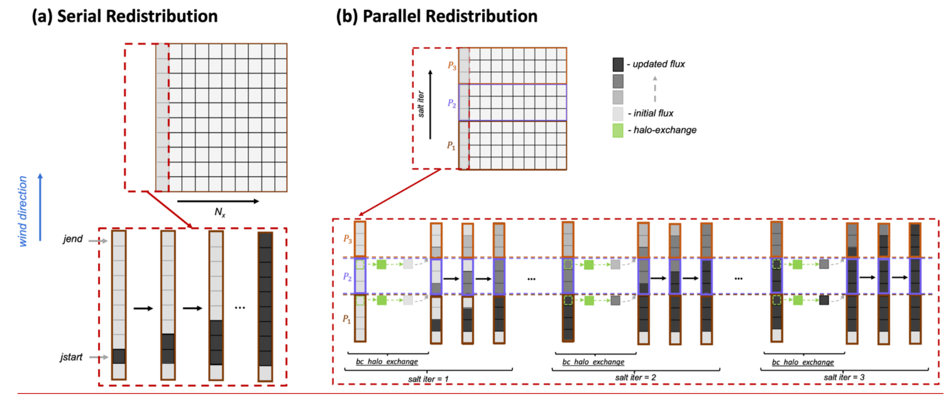


Figure 7: *(a) Schematic of the serial and (b) parallel redistribution algorithm showing the change in saltation flux due to southerly winds over a gridded domain for $N_x = 1$. The parallel schematic demonstrates how three processes (P_1, P_2, P_3) use an additional iteration (`salt iter`) to perform a halo-exchange and update the boundary condition of the saltation flux.*

3.1.3 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 (Figure 8a). This approach requires that one or more processes stores global arrays for input variables and that one process (Process 1; Figure 8a) stores global arrays for all output variables. As the domain size increases, the mapping of local variables to global variables for outputting creates a substantial bottleneck. To improve performance, ***Distributed*** file I/O can be implemented, where input and output files are directly and concurrently accessed by each process (Figure 8b).

Deleted: process's local domain…to another. To minimiz … [16]

Deleted: _

Formatted: Font: Courier New

Formatted: Font: Courier New

Deleted: sublimation. This number was chosen based off … [17]

Deleted: Algorithm 4 in Fig. 7. In the parallel algorithm, the

Formatted: Font: Courier New

Deleted: subroutine performs a halo-exchangeHX…of *Qs* … [18]

Formatted: Font: Courier New

Deleted: process's

```
Algorithm 4 : Parallelized change in saltation flux (southerly wind
initialize boundary condition (Qsalt_v)
do iter = 1, salt_iter
    bc_halo_exchange(Qsalt_v)
    do j = 1, N_x
        do j = jstart, jend
            calculate dQsalt and dh_salt_v
            update Qsalt_v and dh_salt_v !based on vegetation height a
        end do
    end do
end do
```

Deleted:

Deleted: *(Qsalt_v and dh_salt_v) …ue to southerly wind* … [19]

Formatted … [20]

Deleted: *(a), and over three saltation iterations*

Deleted: *for Nx = 1 (b). Before each iteration,…the bou* … [21]

Formatted: Font: Courier New

Formatted: Font: Not Italic

Deleted: 4

Formatted: Heading 3

Deleted: *centralized*

Deleted: (Fig. 8a). However, this approach requires that o … [22]

Deleted: and output variables. As the domain size increas … [23]

Deleted: (Fig. 8b

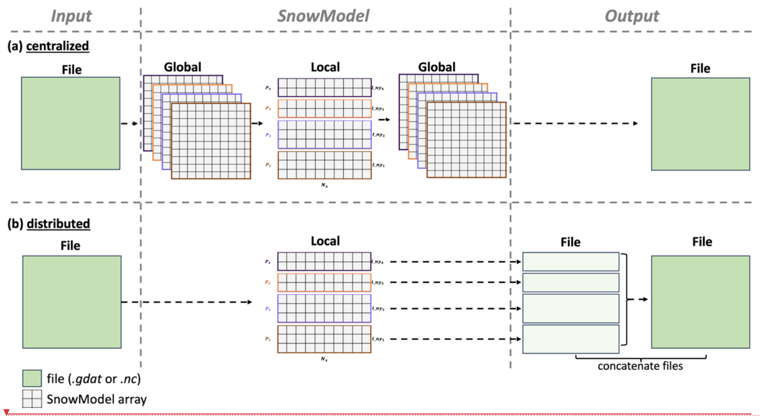


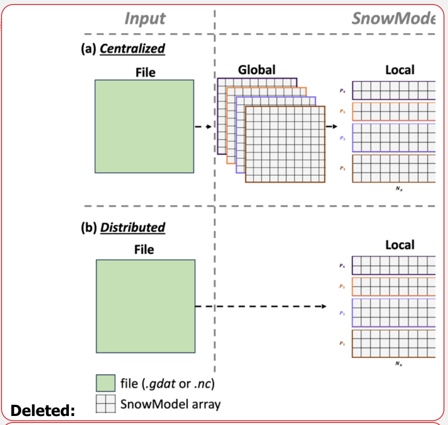
Figure 8: ***(a) Schematic of global to local mapping for file I/O using a Centralized approach with four processes, and (b) Distributed file I/O where each process reads and writes data corresponding to its local domain.***

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). To balance performance and consistency with the serial logic of the code, we used a mixed parallel file I/O approach. A goal of this work was to maintain nearly identical serial and parallel versions of the code in one code base that can be easily maintained and utilized by previous, current, and future SnowModel users with different computational resources and skills. Therefore, we wanted to maintain both the *Centralized* and *Distributed* file I/O approaches. However, for optimal parallel performance over larger simulation domains, 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. This permits the new version of the code to be a drop in replacement for the original serial code without requiring users to install new software libraries or manage hundreds of output files, while enabling users who wish to take advantage of the parallel nature of the code to do so with minimal additional work and no changes to the underlying code.

3.1.3.1 Parallel Inputs

As noted above, SnowModel has two primary types of input files, temporally static files such as vegetation and topography and transient inputs such as meteorological forcing data. While acceptable static input file types include flat binary, NetCDF, and ASCII files for the serial version of the code, optimizing the efficiency of Parallel SnowModel requires static inputs from binary files that can be accessed concurrently and directly subset by indexing the starting byte and length of bytes commensurate to a process local domain. Therefore, each process can read its own portion of the static input data. For very

Deleted:



Formatted: Centered

Deleted: *centralized*

Deleted: *(a),*

Deleted: *distributed*

Deleted: *(b).*

Deleted: In an attempt to

Deleted: File

Deleted: distributed

Deleted: centralized

Deleted: distributed

Deleted: 4

Formatted: Heading 4

Deleted: SnowModel's

Deleted: spatial inputs include

Deleted: vegetation

Deleted: 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

Deleted: and can be read in a centralized or distributed manner. However, since the resolution of static inputs is identical to ... [24]

Deleted: SnowModel's ability to scale to regional and continental sized domains, as storing a single copy of the full domain ... [25]

Deleted: (as noted in the Appendix A), thereby preventing the need for global arrays of static input variables, excessive pr ... [26]

Deleted: process's

Deleted: only reads

large domains, the available memory becomes a limitation when using the centralized approach. For example, the CONUS simulation could not be simulated using a centralized file I/O approach because each process would be holding global arrays of topography and vegetation in memory, each of which would require approximately 5.2 GB of memory per process.

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 grid spacing of the meteorological variables typically ranges from 1 to 30 km and therefore often requires a smaller memory footprint than static inputs for high-resolution simulations. For example, the resolution of NLDAS-2 meteorological forcing has a grid of approximately 11 km, while the high-resolution WRF model used has a 4 km grid. At each timestep, processes read in the forcing data from every station within the domain into a one-dimensional array, index the nearest locations for each SnowModel grid, and interpolate the data to create forcing variables over the local domain. All processes perform the same operation and store common information; 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 and does not impose significant memory limitations for simulations performed to date. While more efficient parallel file input schemes could improve performance, we decided to keep this logic in part to maintain consistency with the serial version of the code and minimize code changes.

3.1.3.2 Parallel Outputs

To eliminate the use of local to global mapping commonly used to output variables (Figure 8a), each process writes its own output file (Figure 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 storage systems making file output using a distributed approach significantly faster than the centralized approach. Therefore, file output in this manner reduces time and memory requirements. Future work could leverage other established parallel I/O libraries at the cost of additional installation requirements.

3.2 Simulation Experiments

Parallel SnowModel experiments were conducted to both evaluate the effectiveness of the parallelization approach used in this study (Sect. 3.1) and to produce a high-resolution snow dataset over CONUS. All experiments were executed with a 100 m grid increment, a 3 h time step, a single-layer snowpack configuration, and included the primary SnowModel modules (MicroMet, EnBal, SnowPack, and SnowTran-3D). These experiments are further described below, with results provided in Sect. 4.

Validation experiments comparing output from the original serial version of the code to the parallel version were conducted continuously throughout the parallel algorithm development to assess the reproducibility of the results. Additionally, a more thorough validation effort was performed at the end of the study that compared output from the serial algorithm to that of the parallel algorithm, while varying the domain size, the number of processes, and therefore the domain decomposition. Results from all of these validation experiments produced root mean squared error (RMSE) values of 10^{-6}, which is at the limit of

Deleted: resolution
Deleted: is
Deleted: much coarser
Deleted: the
Deleted: is
Deleted: a
Deleted: from NCAR is approximately
Deleted: station data
Deleted: nearest station
Deleted: (forcing data stored in the one-dimensional array). However, since the resolution
Deleted: .
Deleted: 4
Formatted: Heading 4
Deleted: (Fig. 8a
Deleted: (Fig. 8b
Deleted: disc
Deleted: 4 Results¶
Deleted: evaluated on six domains in the United States (Fig. 1).
Deleted: and used
Deleted: . 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

machine precision, when compared to serial simulation results. See Appendix B for more details on the validation experiments. The serial version of SnowModel has been evaluated in many studies across different snow classes (Sturm and Liston, 2021; Liston and Sturm, 2021), time periods, and snow properties. Evaluations ranged from snow cover (Pedersen et al., 2016; Randin et al., 2015), snow depth (Szczypta et al., 2013; Wagner et al., 2023), SWE (Freudiger et al., 2017; Hammond et al., 2023; Mortezapour et al., 2020; Voordendag et al., 2021), and SWE-melt (Hoppinen et al., 2023; Lund et al., 2022), using field observations, snow-telemetry stations, and remote sensing products. A full comparison of the Parallel SnowModel simulations presented here with observations across CONUS is beyond the scope of the present work. Incorrectly simulated SWE could affect the scaling results and CONUS visualizations presented in Sect. 3.2.1.1, 3.2.1.2, and 3.2.2; for example, if zero SWE were incorrectly simulated in many locations, processing time would be less than if SWE had been simulated and tracked. However, based on the scale of these analyses and the fact that SnowModel has been previously evaluated in a wide range of locations, we believe the impacts of this limitation on the computational results presented here are minimal.

3.2.1 Parallel Performance

In high performance computing, scalability attempts to assess the effectiveness of running a parallel algorithm with an increasing number of processes. Thus, scalability can be used to identify the optimal number of processes for a fixed domain, understand the limitations of a parallel algorithm as a function of domain size and number of processes, and estimate the efficiency of the parallel algorithm on new domains or computing architectures. Speedup, efficiency, and code profiling were tools used to assess the scalability and performance of Parallel SnowModel on fixed domains. Speedup (S; Eq. 1), a metric of strong scaling, is defined as the ratio of the serial execution time, $T(1)$, over the execution time using N processes, $T(N)$. Optimally, parallel algorithms will experience a doubling of speedup as the number of processes is doubled. Some reasons why parallel algorithms do not follow ideal scaling include the degree of concurrency possible and overhead costs due to communication. Synchronization statements have an associated cost of decreasing the speed and efficiency of an algorithm due to communication overhead and requirements for one process to sit idle while waiting for another to reach the synchronization point. Furthermore, speedup tends to peak or plateau at a certain limit on a given computing architecture and domain because either the overheads grow with an increasing number of processes, or the number of processes exceeds the degree of concurrency inherent in the algorithm (Kumar and Gupta, 1991). For large domains, where serial simulations cannot be performed either due to wall-clock or memory limitations, relative speedup, (S; Eq. 2), is commonly used. Relative speedup is estimated as a ratio of the execution time, $T(P)$, of the minimum number of processes, (P), that can be simulated on a given domain over $T(N)$. An additional speedup metric, approximate speedup (S; Eq. 3), is introduced to estimate S by assuming perfect scaling from P to a single process. While this is only an approximation, it is helpful to compare the S across the different domains on a similar scale. Additionally, efficiency (E; Eq. 4), and approximate efficiency (E; Eq. 5) are the ratios of S to N and S to N, respectively. A simulation that demonstrates ideal scaling, would

Deleted: **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

Deleted: *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.

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

4.2

Deleted:

Formatted: Heading 3

have 100% efficiency. Additionally, code profiling evaluates the cumulative execution time of individual submodules (e.g. Preprocess, Readparam, MicroMet, Enbal, SnowPack, SnowTran-3D, and Output) 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 enhance performance.

$$\boldsymbol{S(N) = \frac{T(1)}{T(N)}} \qquad \text{Eq. 1}$$

$$\boldsymbol{S(N) = \frac{T(P)}{T(N)}} \qquad \text{Eq. 2}$$

$$\boldsymbol{S(N) = \frac{T(P)}{T(N)} * P} \qquad \text{Eq. 3}$$

$$\boldsymbol{E(N) = \frac{S}{N} * 100\%} \qquad \text{Eq. 4}$$

$$\boldsymbol{E(N) = \frac{S}{N} * 100\%} \qquad \text{Eq. 5}$$

3.2.1.1 Parallel Improvement

To better understand how changes to the Parallel SnowModel code have affected its performance, speedup and code profiling plots were assessed for simulations using three distinct versions of the code. These versions represent snapshots of the algorithms development and quantify the contributions of different types of code modifications to the final performance of the model. These versions were identified by different GitHub commits (Mower et al., 2023) and can be summarized as follows. The first or baseline version represents an early commit of Parallel SnowModel, where file I/O is performed in a *Centralized* way, as described in Sect. 3.1.3. Each process stores both a local and global array in memory for all input variables, makes updates to its local arrays, distributes that updated information into global arrays used by one process to write each output variable. The embarrassingly parallel portion of the physics code has been parallelized, but the snow redistribution step is not efficiently parallelized, it has a larger number of synchronizations and memory transfers. Therefore, this approach has significant time and memory constraints. The *Distributed* version represents an instance of the code where distributed file I/O (Sect. 3.1.3) had first been implemented. In this version, each process reads and writes input and output variables for its local domain only. Global arrays and the communication required to update these variables are no longer needed; this alleviates memory constraints and shows the value of parallelizing I/O in scientific applications. Lastly, the *Final* version represents the most recent version of Parallel SnowModel, (at the time of this publication) where the snow transport algorithm had been optimized to run efficiently. This was done by reducing unnecessary memory allocations, reducing the transfer of data via coarrays, and optimizing memory transfers to reduce synchronization calls. This shows the value of focused development on a single hotspot of the code base. The simulations were executed on the CO Headwaters domain (Figure 2) using 1, 2, 4, 16, 36, 52, 108, and 144 processes, outputted only a single variable, and were forced with

NLDAS-2 data from 23-24 March 2018. While 2-days is a short period to perform scaling experiments, a significant amount of wind and frozen precipitation was observed over the CO Headwaters domain during the simulation to activate some of the snow redistribution schemes in SnowTran-3D. Furthermore, to avoid disproportionately weighing the initialization of the algorithm, we removed the timing values from the ReadParam and Preprocess submodules from the total execution time used in the speedup analysis. Results from these experiments are provided in Sect. 4.1.

3.2.1.2 Strong Scaling

Strong scaling experiments of Parallel SnowModel were evaluated by comparing the approximate speedup and efficiency (`S` and `E`) over six different size domains across the United States, all with a 100 m grid spacing [Tuolumne, CO Headwaters, Idaho, PNW, Western U.S., and CONUS] (Figure 2). These experiments use the *Final* version of the code according to Sect. 3.2.1.1. The simulations were forced with NLDAS-2 data for 2928 timesteps from 1 September 2017 to 1 September 2018 and output one variable (SWE). The number of processes used in these simulations varied by domain based on the 12 h wall-clock and memory constraints on Cheyenne. Results from these experiments are provided in Sect. 4.2.

3.2.2 CONUS Simulations

A primary goal of this work was to run Parallel SnowModel simulations for 21 years (2000 – 2021) over the CONUS domain (Figure 2) 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, melt rate, and sublimation. Future work will analyze results from these simulations. The CONUS 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.

4 Results

4.1 Parallel Improvement

Figure 9 demonstrates how the scalability of Parallel SnowModel evolved, as shown through code profiling (top row; Figure 9) and speedup (bottom row; Figure 9) plots at three different stages (*Centralized*, *Distributed*, and *Final*) of the code development. The code profiling plots display the cumulative execution time of each submodule on a logarithmic scale as a function of the `N`. The strong scaling plots show the total execution time (`T(N)`) and the speedup (`S(N)`; Eq. 1) as a function of `N` on the primary y-axis and secondary y-axis, respectively. As mentioned previously, the initialization timing was removed from these values. The speedup of the *Centralized* version of the code quickly plateaus at approximately 10 processes. While the Enbal, SnowPack, and MicroMet subroutines scale with the number of processes (execution time decreases proportional to the increase in the number of processes), the ReadParam, Preprocess, and Output subroutines, which all perform file I/O or memory allocation, require a fixed execution time regardless of the number of processes used, and the execution time of the SnowTran-3D submodule increases beyond 16 processes. This highlights the large bottleneck that often occurs during the file I/O step in scientific code and the importance of code infrastructure outside of the physics routines. In contrast, all of the submodules in the *Distributed* version of the code, scale up to 36 processes, at which point the inefficient parallelization of the SnowTran-3D submodule causes a significant slowdown, an increase in execution time as the number of processes increases. This results in a speedup that plateaus at 52 processes and decreases beyond 108 processes. In the *Final* version of the code, scalability is observed well beyond 36 processes, with a maximum speedup of 100 observed using 144 processes. The execution time of all the submodules decreases as the number of processors increases. This work highlights the value of going beyond the rudimentary parallelization of a scientific code base by profiling and identifying individual elements that would benefit the most from additional optimization. This is a well-known best practice in software engineering but often underappreciated in high-performance scientific computing. In Parallel SnowModel, the improvement of these communication bottlenecks is primarily attributed to utilizing a distributed file I/O scheme and minimizing processor communication by limiting the use of coarrays and synchronization calls. Ultimately, without these improvements, the CONUS domain could not be simulated using Parallel SnowModel.

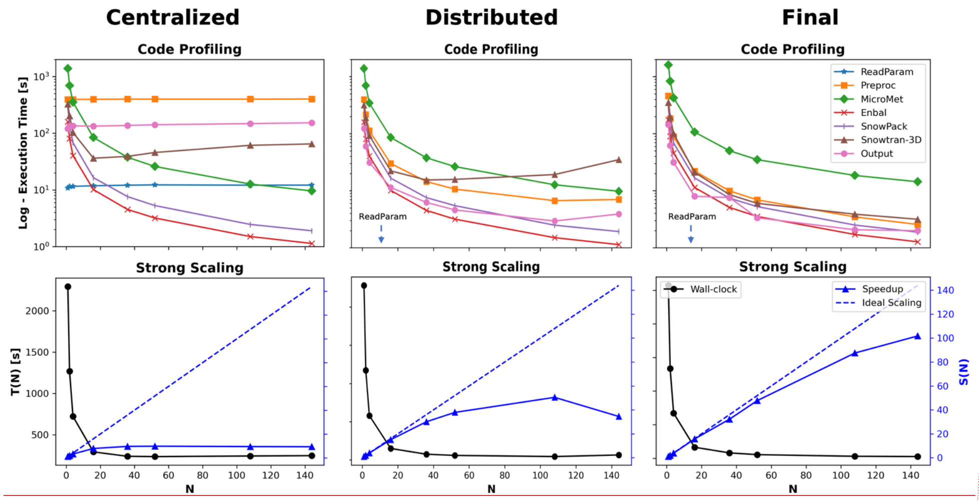


Figure 9: Code profiling (top row) and strong scaling (bottom row) results demonstrating the progression of Parallel SnowModel, which includes a version of the code with centralized file I/O (*Centralized*; first column), a version of the code with distributed file I/O (*Distributed*; second column), and a final version of the code at the time of this publication (*Final*; third column). These versions can be found as different commits within the GitHub repository (Mower et al., 2023). The code profiling plots display the cumulative execution time of each submodule on a logarithmic scale as a function of the number of processes (N). The arrow in the code profiling plots of *Distributed* and *Final* indicates the ReadParam timing is below the y-axis at approximately 0.3 seconds and

Deleted: 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. Conv ... [27]

Deleted: and high process communication

Deleted: *High Sync*

Deleted: more recent

Deleted: which includes distributed file I/O and low process communication (*Distributed Low Sync*…

Formatted: Font: Italic

Deleted: *High Sync* and *Distributed Low Sync*

Formatted: Font: Not Italic

0.003 seconds, respectively. The strong scaling plots show the total execution time ($T(N)$) against N on the primary y-axis and the speedup (S) against N on the secondary y-axis.

4.2 Strong Scaling

In addition to the parallel improvement analysis, strong scaling was also performed on six domains for the 2018 water year to better understand how Parallel SnowModel scales across different domain sizes and decompositions. Figure 11 displays the approximate speedup (S; Eq. 3) of Parallel SnowModel for three local/state domains (Tuolumne, CO Headwaters, and Idaho) and three regional/continental domains (PNW, Western US, and CONUS). Additionally, Table 1 contains information about the minimum and maximum number of processors (P and P^*, respectively) simulated on each domain and their corresponding execution time, relative speedup (S; Eq. 2), approximate S across the different domains on a similar scale.

$$S(N) = \frac{T(P)}{T(N)} * P \quad (5)$$

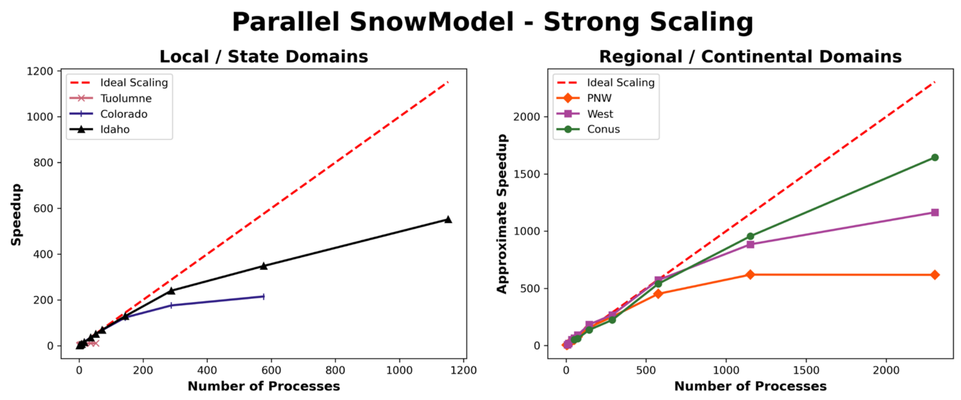


Figure 10: *The left panel displays* speedup (S; Eq. 3), and approximate efficiency (E; Eq. 5). As mentioned previously, simulations were constrained by both the 12 h wall-clock and 109 GB of memory per node on the Cheyenne supercomputer.

In strong scaling, the number of processes is increased while the problem size remains constant; therefore, it represents a reduced workload per process. Local-sized domains, e.g., Tuolumne, likely do not warrant the need for parallel resources because they have small serial runtimes (e.g., using 52 processes, Tuolumne had an E of 38%; Table 1). However, state, regional, and continental domains stand to benefit more significantly from parallelization. The CONUS runtime decreased by a factor of 2.6 running on 3456 processes relative to 648 processes. Based on our approximate speedup assumption, we would estimate a CONUS S of 1690 times on 3456 processes compared to one process, with an E of 49%. The Western US and PNW domains display very similar scalability results (Figure 11), which is attributed to the similar number of grid cells

Deleted: performing a

Deleted: analysis across different versions of the code, we

Deleted: a scaling experiment across several

Deleted: using

Deleted: current version of

Deleted: (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),…

Deleted: S) is estimated using the execution time, $T(P)$, of the simulation with the minimum number of processes (P) by assuming perfect scaling from there to a single process (Eq. 5). For example, this experiment identified the P needed to run the PNW don ... [28]

Deleted: one core would be 416 min. Near perfect scaling ... [29]

Deleted:) appears to be justified. While this approximatic ... [30]

Deleted: *Eq. 4) for local and state sized simulations (Tuo* ... [31]

Formatted: Font: Not Italic, English (UK)

Formatted: Font: Not Italic, English (UK)

Deleted: *5) for the regional and continental sized domai* ... [32]

Deleted: Strong scaling analysis is useful for I/O and men ... [33]

Deleted: and parallel efficiencies (E; Eq. 6), or approxima ... [34]

Deleted: a E of 20%).

Deleted: 32

Deleted: 2304

Deleted: 52

Deleted: expect

Deleted: S

Deleted: 1644

Deleted: 2304

Deleted: core, with an E of 71%. If the initialization porti ... [35]

Deleted: initialization process, where the process count al ... [36]

Formatted: Font color: Black

in the y dimension (Figure 2; and Table 1) and thus parallel decomposition for each domain. Furthermore, these domains may also have a similar proportion of snow-covered grid cells. While the PNW likely has more terrestrial grid cells that are covered by snow for a longer period throughout the water year, it also has a significant number of ocean grid cells where snow redistribution would not be activated.

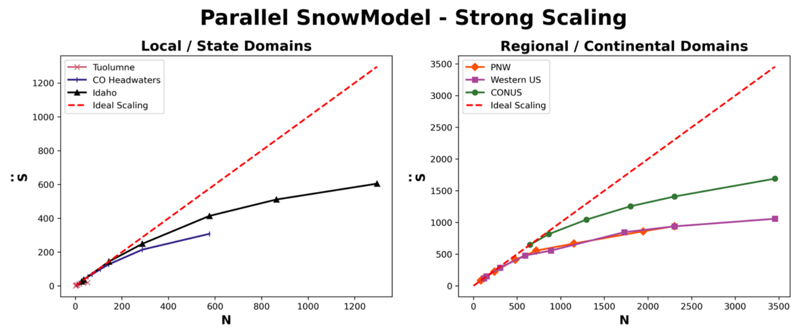


Figure 11: The left panel displays approximate speedup (S; Eq. 3) as a function of the number of processes (N) for local and state sized simulations (Tuolumne, CO Headwaters, and Idaho), while the right panel shows S for the regional and continental sized domains (PNW, Western US, and CONUS).

Domain	Nx	Ny	P̂ or P*	Number of Processes	Execution Time (m)	Relative Speedup	Approximate Speedup	Approximate Efficiency (%)
				N	$T(N)$	$\hat{S}(N)$	$\bar{S}(N)$	$\bar{E}(N)$
Tuolumne	311	185	P̂	1	13	---	---	100
			P*	52	0.7	20	20	38
CO Headwaters	3166	5167	P̂	8	934	---	8	100
			P*	576	24	39	308	53
Idaho	6916	9107	P̂	27	1068	---	27	100
			P*	1296	48	22	605	47
PNW	13677	16058	P̂	84	1173	---	84	100
			P*	2304	105	11	941	41
Western US	17737	17878	P̂	120	1187	---	120	100
			P*	3456	135	9	1058	31
CONUS	46238	28260	P̂	648	1196	---	648	100
			P*	3456	459	3	1690	49

Table 1: Strong scaling results containing grid dimensions (N_x and N_y), number of processes, execution time, relative speedup, approximate speedup, and approximate efficiency of simulations executed with the minimum and maximum number of processes for the Tuolumne, CO Headwaters, Idaho, PNW, Western US, and CONUS domains.

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. Based on these scaling results, Figure 12 contains the relationship between the number of processes (N) at which each domain is estimated to reach 50% E (using linear interpolation) with the total number of grid cells in the y dimension (N_y) and the average number of grid cells in the y dimension per process (l_{ny}; inset Figure 12). At this level of efficiency, it is notable the consistency of both the linear relationship between N_y and N (8.7:1 ratio) and the values of l_{ny} (5 to 11) for these year-long simulations that vary in both domain size and the proportion of snow-covered area. Similar relationships (Figure 12) can be used to approximate the scalability of Parallel SnowModel on different sized domains and can be adjusted for the desired level of efficiency. For example, we decided to run the CONUS simulations (Sect. 4.3) using 1800 processes based on its 70% approximate efficiency.

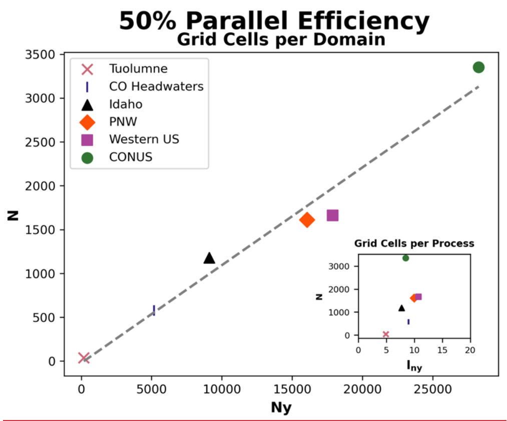


Figure 12: Relationship between the number of grid cells in the y dimension (Ny) and the number of processes (N) for each domain at which 50% approximate efficiency is estimated using the strong scaling analysis. The dashed line represents the best fit line for this relationship using OLS regression. The inset figure displays a similar relationship but compares N to the average number of grid cells in the y dimension per process (l_{ny}), instead of Ny.

4.3 CONUS Simulations

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 Figure 13. On this date, simulated SWE was observed throughout the northern portion of the CONUS domain with the largest values concentrated in the mountain ranges (Figure 13a). The Indian Peaks sub-domains of distributed SWE (Figure 13b) with reference topography (Figure 13c) underscores the ability of the large dataset to capture snow processes in a local alpine environment. It is important to note that while SnowModel does simulate snow redistribution, it does not currently have an avalanche model, which may be a limitation of accurately simulating SWE within this sub-domain. Additionally, Figure 13b highlights two grid cells located 200 m apart on a peak. Figures 11d and 11e display the SWE evolution of these two grid cells over the entire dataset (water years 2000 – 2021) and the 2011 water year, respectively, further demonstrating 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 the 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.

Deleted: *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. ...

Deleted: Fig. 11. The

Deleted: domain

Deleted: The time series of

Deleted: for those grid cells (Fig. 11d and Fig. 11e) demonstrates

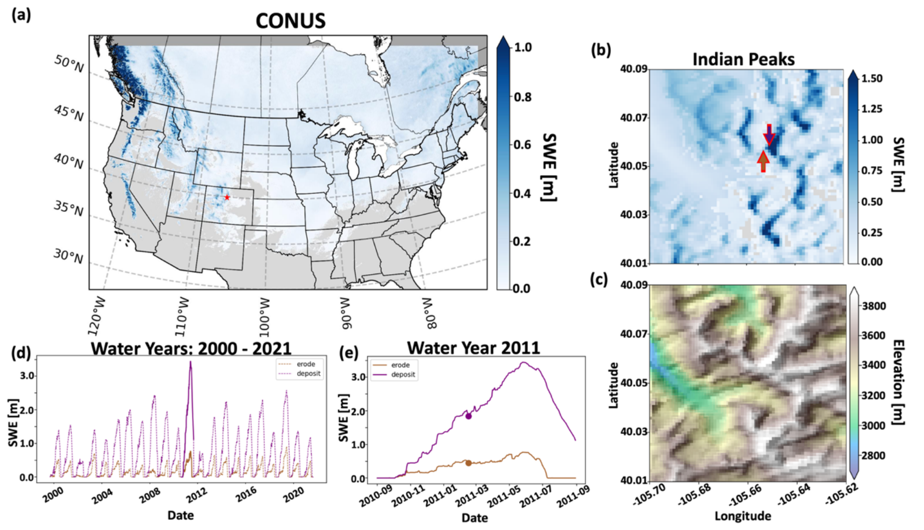
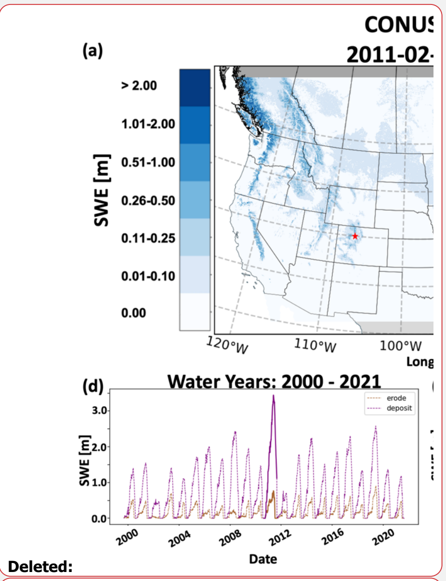


Figure 13: ***Simulation results of Parallel SnowModel over CONUS using the WRF projection. (a) Spatial patterns of SWE over the CONUS domain for 12 February 2011, (b) highlighting the SWE distribution (c) and topography with an applied hillshade of a sub-domain near Apache Peak in the Indian Peaks west of Boulder, CO. (d) Time series of SWE from 2000-2021 and (e) over the 2011 water year for grid cells ("erode" and "deposit") identified in panel (b). 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

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 (Figure 11), suggests that Parallel SnowModel scales effectively over regional to continental domains using the one-dimensional decomposition 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 implemented 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 algorithm versions and across different domains sizes, and demonstrate its ability to simulate continental domains at high-resolution. Code profiling and speedup analyses over the CO Headwaters domain helped identify bottlenecks in file I/O and processor communication in SnowTran-3D during the development of the parallel algorithm



Deleted:

Formatted: Font: Italic, English (US)

Deleted: *(a), highlighting the SWE distribution (cb… ar* ... [37]

Formatted: Font: Not Italic, English (UK)

Moved down [5]: 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

Moved down [6]: 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

Deleted: In this paper, we present a relatively simple

Deleted: 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

Deleted: HX coarrays for the wind and solar radiation models, 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. ¶

Formatted: Font: Courier New, Not Italic

Deleted: of Parallel SnowModel on high-performance cor ... [38]

Deleted: versions of algorithm versions and across differe ... [39]

(Sect. 4.1). Corrections to the referred bottlenecks allowed Parallel SnowModel to scale up to regional and continental sized simulations and highlights the value of optimizing scientific code. The scalability analyses showed the effectiveness of running Parallel SnowModel with an increasing number of processes on state, regional, and continental domains that contain different proportions in both size (Nx and Ny) and snow-covered grid cells (Sect 4.2). For Parallel SnowModel scalability is primarily dependent on the number of grid cells per process (Nx and l_{ny}) and is affected by snow redistribution, which is dependent on the proportion of terrestrial grid cells with sufficient winds and available soft snow to be redistributed (Sect. 3.1.2.2). For example, a maritime snowpack (e.g. PNW) as compared to a continental snowpack (e.g. CO Headwaters), may be deeper and more spatially extensive but potentially lacks a high frequency of soft snow above tree-line to activate snow redistribution. Furthermore, the similar relationships among efficiency and domain decomposition observed on the simulated domains that vary in size, topography, vegetation, and snow classes (Sturm and Liston, 2021; Liston and Sturm, 2021) (Fig. 10), make it reasonable to extrapolate the results from these simulation experiments to other domains within CONUS.

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) 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 (Figure 11), we estimate it would take over 2.5 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. Therefore, by the time it took the large server to complete a CONUS simulation for one water year, 12 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 (Sect. 4.3).

SnowModel can simulate 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. While several snow data products exist, few capture the suite of snow properties along with the spatio-temporal extents and resolutions that can benefit a wide variety of applications. For example, current snow information products include the 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 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 has limited spatio-temporal coverage and a high cost of acquisition. Furthermore, there are many

Deleted: 2

Deleted: . Parallel speedup

Deleted: helped to identify the optimum

Formatted: Font color: Auto

Deleted: of the parallel algorithm for different

Formatted: Font color: Auto

Formatted: Font color: Auto

Deleted: sizes (Sect 4.2).

Deleted: 1) and processes (52; Fig. 10

Deleted: (Fig. 10),

Deleted: 10

Deleted: (Sect. 4).

Deleted: 48

Deleted: .

Moved (insertion) [7]

Moved (insertion) [8]

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. The new Parallel SnowModel described here permits the application of this modeling system to very large domains without sacrificing spatial resolution.

6 Conclusions

In this paper, we present a relatively simple parallelization 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.) was parallelized and modularized in this study. This allows SnowModel to be compiled with a range of Fortran compilers, including 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 halo-exchange coarrays for the wind and solar radiation models, as well as for snow redistribution. The scalability of Parallel SnowModel was demonstrated over different sized domains, and the new code enables the creation of high-resolution simulated snow datasets on continental scales. This parallelization approach can be adopted in other parallelization efforts where spatial derivatives are calculated or fluxes are transported across gridded domains.

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 CO 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

Moved (insertion) [9]

Moved up [7]: 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-

Moved up [8]: . 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

Moved up [9]: The new Parallel SnowModel described here permits the application of this modeling system to very large domains without sacrificing spatial resolution.¶

Moved (insertion) [5]

Deleted: 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 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.

Deleted: is flown on an aircraft and thus has limited spatio-temporal coverage. Furthermore, there are many fields of study that

Deleted: 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

Moved (insertion) [6]

Formatted: Font: Courier New, Not Italic

Deleted: lagrangian

Deleted: Colorado

Deleted:

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 temperature [$°C$], wind direction [°], and wind speed [$m * s^{-1}$]. Ultimately, we feel confident that Parallel SnowModel is producing the same results as the original serial algorithm.

Deleted: 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

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). For more information about the serial version of SnowModel, refer to Liston and Elder (2006a). 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.

Deleted: The code base is limited to the parallelization changes to

Deleted: 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

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

This material is based upon work supported by the NSF National Center for Atmospheric Research, which is a major facility sponsored by the U.S. National Science Foundation under Cooperative Agreement No. 1852977. 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), support by the University of Washington's College of Engineering Fellowship, and computational support from NSF NCAR Computational and Information Systems

Deleted:) and

Lab (CISL) and NASA High-End Computing (HEC) Program through the NASA Center for Climate Simulation (NCCS) at Goddard Space Flight Center.

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

Deleted: ¶

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.

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

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. J., Casey, K. A., and Choudhury, B.: A blended global snow product using visible, passive microwave and scatterometer satellite data, International Journal of Remote Sensing, 32, 1371-1395, 10.1080/01431160903548013, 2011.

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

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

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

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

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

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

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

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

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.

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.

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.

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

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

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

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

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

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

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

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

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

Page 7: [1] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 7: [2] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 7: [3] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 8: [4] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 8: [4] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 8: [4] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 8: [5] Formatted Ross Mower 1/31/24 11:16:00 PM

Font: Courier New, Not Italic

Page 8: [5] Formatted Ross Mower 1/31/24 11:16:00 PM

Font: Courier New, Not Italic

Page 8: [6] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 8: [6] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 8: [6] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 8: [6] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 8: [6] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 8: [6] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 8: [6] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 9: [7] Deleted **Ross Mower** **1/31/24 11:16:00 PM**

Page 9: [8] Deleted **Ross Mower** **1/31/24 11:16:00 PM**

Page 10: [9] Deleted **Ross Mower** **1/31/24 11:16:00 PM**

Page 10: [9] Deleted **Ross Mower** **1/31/24 11:16:00 PM**

Page 10: [10] Deleted **Ross Mower** **1/31/24 11:16:00 PM**

Page 10: [10] Deleted **Ross Mower** **1/31/24 11:16:00 PM**

Page 10: [11] Deleted **Ross Mower** **1/31/24 11:16:00 PM**

Page 10: [11] Deleted **Ross Mower** **1/31/24 11:16:00 PM**

Page 10: [12] Deleted **Ross Mower** **1/31/24 11:16:00 PM**

Page 10: [12] Deleted **Ross Mower** **1/31/24 11:16:00 PM**

Page 10: [12] Deleted **Ross Mower** **1/31/24 11:16:00 PM**

Page 10: [12] Deleted **Ross Mower** **1/31/24 11:16:00 PM**

Page 10: [12] Deleted **Ross Mower** **1/31/24 11:16:00 PM**

Page 10: [12] Deleted **Ross Mower** **1/31/24 11:16:00 PM**

Page 10: [13] Formatted **Ross Mower** **1/31/24 11:16:00 PM**

Font: Courier New, Not Italic

Page 10: [13] Formatted Ross Mower 1/31/24 11:16:00 PM

Font: Courier New, Not Italic

Page 10: [14] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 10: [14] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 10: [14] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 10: [14] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 10: [14] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 10: [14] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 10: [15] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 10: [15] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 11: [16] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 11: [16] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 11: [17] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 11: [17] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 11: [18] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 11: [18] Deleted Ross Mower 1/31/24 11:16:00 PM

**Page 11: [18] Deleted	Ross Mower	1/31/24 11:16:00 PM**

**Page 11: [19] Deleted	Ross Mower	1/31/24 11:16:00 PM**

**Page 11: [19] Deleted	Ross Mower	1/31/24 11:16:00 PM**

**Page 11: [20] Formatted	Ross Mower	1/31/24 11:16:00 PM**

Font: Courier New

**Page 11: [20] Formatted	Ross Mower	1/31/24 11:16:00 PM**

Font: Courier New

**Page 11: [20] Formatted	Ross Mower	1/31/24 11:16:00 PM**

Font: Courier New

**Page 11: [21] Deleted	Ross Mower	1/31/24 11:16:00 PM**

**Page 11: [21] Deleted	Ross Mower	1/31/24 11:16:00 PM**

**Page 11: [22] Deleted	Ross Mower	1/31/24 11:16:00 PM**

**Page 11: [22] Deleted	Ross Mower	1/31/24 11:16:00 PM**

**Page 11: [23] Deleted	Ross Mower	1/31/24 11:16:00 PM**

**Page 11: [23] Deleted	Ross Mower	1/31/24 11:16:00 PM**

**Page 11: [23] Deleted	Ross Mower	1/31/24 11:16:00 PM**

**Page 11: [23] Deleted	Ross Mower	1/31/24 11:16:00 PM**

**Page 11: [23] Deleted	Ross Mower	1/31/24 11:16:00 PM**

Page 12: [24] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 12: [25] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 12: [26] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 17: [27] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 18: [28] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 18: [29] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 18: [30] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 18: [31] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 18: [32] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 18: [33] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 18: [34] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 18: [35] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 18: [36] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 21: [37] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 21: [37] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 21: [37] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 21: [37] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 21: [38] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 21: [38] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 21: [38] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 21: [38] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 21: [38] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 21: [39] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 21: [39] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 21: [39] Deleted Ross Mower 1/31/24 11:16:00 PM

Page 21: [39] Deleted Ross Mower 1/31/24 11:16:00 PM