Flexible User-defined Domain Decomposition in Kilometer-Scale E3SM Land Model Simulation

Zhuowei Gu[1], Dali Wang[2], Dawei Gao[3], Yunhe Feng[3], and Qinglei Cao[1]

[1] Saint Louis University, USA
[2] Oak Ridge National Laboratory, USA
[3] University of North Texas, USA

Abstract. The Energy Exascale Earth System Model (E3SM) Land Model (ELM) has been extended to kilometer-scale (km-ELM) resolutions, enabling high-fidelity simulations of terrestrial processes at 1 km × 1 km grid spacing. In ELM, domain decomposition partitions the computational domain across processors, ensuring efficient parallel execution. Currently, round-robin decomposition is applied, providing a straightforward way to distribute computational workload. As ELM continues evolving at the kilometer-scale (km-scale), particularly with integrating lateral flow modeling, decomposition strategies must also account for the increased workload and data movement. This paper introduces a flexible user-defined domain decomposition framework, allowing users to customize domain partitioning based on application requirements. The impact of different decomposition strategies is evaluated across various applications concerning computation, communication, and I/O. Results demonstrate that while 1D partitioning yields superior I/O performance, k-nearest neighbors (KNN) clustering effectively reduces interprocess communication overhead. This study lays the groundwork for scalable partitioning in large-scale land surface simulations, enhancing next-generation Earth system modeling.

Keywords: Earth System Modeling, E3SM Land Model (ELM) · Kilometer-Scale ELM · Domain Decomposition · Load Balancing · I/O

1 Introduction

The Energy Exascale Earth System Model (E3SM) [1] is a state-of-the-art, high-resolution Earth system model developed to simulate and project climate change with a focus on water cycle, biogeochemical, and cryospheric processes. Designed for exascale computing, E3SM integrates multiple components, including atmosphere, ocean, land, and ice, to provide high-fidelity climate predictions. The land component of E3SM, known as the E3SM Land Model (ELM), is responsible for simulating terrestrial processes such as energy balance, water fluxes, vegetation dynamics, and biogeochemical cycles. ELM has evolved as a sophisticated land surface model, supporting simulations at various spatial resolutions and facilitating studies on climate-land interactions.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97626-1_12

With the advancement of high-performance computing (HPC) and the increasing demand for more granular climate simulations, ELM has been extended to kilometer-scale ELM (km-ELM) [2,3]. Unlike traditional land models that operate at coarser resolutions (e.g., 10–100 km), km-ELM enables fine-scale simulations down to 1 km $\times$ 1 km grid spacing, capturing small-scale heterogeneities in land surface processes. This capability is crucial for improving regional climate assessments, hydrological modeling, and ecosystem studies, particularly in complex terrains and highly heterogeneous landscapes.

However, achieving efficient and scalable kilometer-scale (km-scale) simulations poses significant challenges due to the sheer volume of data and computational workload. First, the computational cost increases drastically as the number of grid cells grows, requiring efficient load balancing to prevent idle processors and ensure optimal utilization of computing resources. Second, km-ELM requires lateral flow processes. Unlike traditional column-based land models where each grid cell operates independently, lateral flow requires frequent data exchanges between neighboring subdomains to simulate surface and subsurface water transport. Third, the high-resolution nature leads to substantial I/O demands, particularly in restart files and history files [3].

These challenges can cause potential bottlenecks in km-ELM in load balancing, data movement, and I/O, especially when considering the current simple approach of round-robin decomposition strategy [4, 5]. Efficient domain decomposition strategies, a technique used to partition the vast computational domain into smaller subdomains for parallel processing, are critical to ensuring that computational workloads are evenly distributed across processors while minimizing communication overhead, allowing km-ELM to leverage modern supercomputing architectures effectively.

In this paper, we introduce a flexible user-defined domain decomposition framework (FUDD) in km-scale ELM, allowing users to customize domain partitioning based on specific application requirements. Unlike the current round-robin strategy, FUDD provides adaptability, enabling optimized load balancing, efficient communication, and improved I/O performance across different km-ELM use cases. The contributions of this paper are as follows:

- Introducing a flexible user-defined domain decomposition framework integrated into km-scale ELM.
- Demonstrating domain decomposition using static predefined strategies, adaptive data-driven methods, and watershed-based partitioning.
- Evaluating decomposition strategies' impact in multiple applications on different clusters in terms of computation, communication, and I/O.

To the best of our knowledge, this is the first work to systematically study the effect of domain decomposition in kilometer-scale ELM simulations.

The structure of this paper is as follows. Section 2 discusses prior research relevant to our work. Section 3 provides the necessary background on km-scale ELM along with domain partitioning and gridcell grouping. The user-defined decomposition framework is described in Section 4. A performance evaluation

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97626-1_12

and analysis is then conducted in Section 5, and finally, Section 6 outlines our conclusions and future directions.

2 Related Work

2.1 Km-scale E3SM Land Model Development

E3SM [1] is a state-of-the-art climate modeling system built to capture intricate interactions among atmospheric, oceanic, terrestrial, and cryospheric processes with fine-grained spatial and temporal fidelity. Within this framework, ELM plays a pivotal role by offering a sophisticated representation of surface processes, including energy fluxes, hydrological cycles, vegetation dynamics, and biogeochemical transformations [6].

The ELM codebase is extensive, comprising nearly half a million lines of code. It employs a diverse range of specialized data structures, including roughly 2,000 globally referenced multidimensional arrays and more than 1,000 subroutines [7,8]. In response to the increasing demand for precision in land surface simulations, researchers have recently introduced ultrahigh-resolution ELM (uELM) models. These leverage Exascale computing capabilities to conduct highly detailed simulations at continental and global scales. Cutting-edge computational frameworks have been devised to optimize uELM execution on hybrid Exascale platforms, further enhancing the efficiency and accuracy of these large-scale simulations [2, 9]. In one notable study from 2022, a high-resolution ELM simulation at a 1 km $\times$ 1 km scale was applied to such a limited domain [10]. This investigation utilized a streamlined ELM model, incorporating a reduced number of subgrid components alongside a satellite phenology submodel, aiming to examine subgrid topographic influences on land-atmosphere interactions in mountainous regions. The increasing availability of high-resolution datasets, including detailed climate forcing and soil property information [11–13], has significantly improved the feasibility of conducting km-scale ELM simulations. These datasets provide the necessary spatial resolution to better capture complex geographical features and extreme meteorological phenomena. The ongoing development of a kilometer-scale (km-scale) ELM is being pursued alongside other E3SM components, including high-resolution atmosphere and ocean models, to ensure optimal performance on Exascale systems [14, 15].

Despite these advancements, transitioning to high-resolution simulations can present substantial computational challenges. The increased grid density significantly amplifies resource demands, necessitating robust computing power and storage solutions to accommodate the expansive data volumes.

2.2 Domain Decomposition in E3SM

Domain decomposition is a critical technique for improving the scalability and efficiency of land surface models, particularly in high-resolution Earth system simulations. The scalability of land models in Earth system simulations has been

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97626-1_12

a long-standing challenge. Hoffman et al. [4] analyzed parallel efficiency and domain decomposition strategies for climate models, emphasizing the importance of load balancing in large-scale simulations. More recent work has further optimized land model performance at high resolutions. Tang et al. [5] introduced a km-scale ELM framework for large-domain simulations, demonstrating the scalability of ELM at continental scales. They employed a basic round-robin domain decomposition, which, while effective in load distribution, presents inefficiencies when lateral interactions, such as hydrological processes, are introduced. However, existing decomposition strategies, such as round-robin, often struggle with heterogeneous land cover and variable workloads [3]. While these approaches ensure equal partitioning, they do not account for communication costs associated with lateral flow or optimize I/O performance for large-scale km-ELM simulations. Our work introduces a flexible user-defined domain decomposition approach, allowing users to tailor partitioning strategies based on specific application needs, addressing both load balancing and communication overhead.

3 Background

3.1 Km-scale ELM

This work utilizes ELM in *land-only mode*, isolating it from atmospheric feedback mechanisms. Instead of interactive coupling, it employs observational datasets to impose atmospheric forcing, enabling the simulation of ecosystem dynamics under past climatic conditions [16–19]. By adopting this configuration, the study ensures that ecosystem responses to climate fluctuations are examined independently, free from interactions with other components of the Earth system.

ELM functions as a terrestrial ecosystem model driven by data, where each gridcell is processed independently. Its execution requires numerous computational cycles spanning gridcells and their respective subgrid elements. Though the model consists of more than 1000 subroutines, none impose a significant computational burden. The model advances in half-hourly or hourly increments. At every step, computations are performed over each *clump*, a structured grouping of gridcells and their active subgrid components, to update ecosystem state variables and flux exchanges among terrestrial processes. This process relies on more than 3,000 global arrays to manage state variables and nutrient cycles. While most calculations related to subgrid and gridcell dynamics—such as the nutrient cycles of carbon (C), nitrogen (N), and phosphorus (P)—are conducted independently, certain operations introduce dependencies.

ELM's computational demands stem from handling extensive global datasets while rigorously upholding the conservation principles of mass and energy. To maintain these conservation laws, ELM systematically compiles all pertinent state variables and rigorously verifies their consistency at every timestep throughout all gridcells. At the end of a simulation, ELM produces a substantial volume of output data. By default, each monthly history file records more than 550 variables, encapsulating diverse terrestrial ecosystem dynamics such as hydro-

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97626-1_12

logical fluxes, energy exchanges, and the biogeochemical interactions of carbon, nitrogen, and phosphorus across all land grid cells.

3.2 Domain Partitioning and Gridcell Grouping

At the beginning of a simulation, the ELM model systematically traverses the computational domain, assigning unique identifiers to individual land grid cells. These cells are then distributed among MPI (Message Passing Interface) processes following a round-robin assignment approach, which helps maintain an even computational load across processes [4]. Within each MPI process, a collection of specialized datatypes is utilized to delineate the computational domain (see Table 1). After distribution, the assigned grid cells are further organized into structures referred to as *clumps* within each MPI process. Every clump maintains metadata, including the MPI rank of the corresponding processor and information about the total count of subgrid components, along with their respective index boundaries. To optimize memory usage, each MPI process preallocates contiguous memory blocks (arrays) to store ELM variables corresponding to its assigned grid cells and their subgrid components. ELM variables are globally allocated and initialized as arrays to facilitate uniform access throughout the execution. Each grid cell is provisioned with a predefined upper limit on the number of subgrid components, and during each timestep, a filtering mechanism identifies and tracks the active subgrid components within individual grid cells.

Table 1: Customized Data Structures for Defining the Computational Domain

Datatype	**Description**
`npes`	Represents the total number of MPI processes used in the simulation.
`nclumps`	Represents the total number of clumps in the simulation.
`begg & endg`	Determine offset for each clump assigned to each process
`clumps`	Contains the owner process ID, size of grid cells (including subgrid components), and the starting and ending indices of these subgrid elements within each clump. Each process can have multiple clumps.
`lcid`	A mapping structure used to store the processor ID
`gdc2glo`	1D mapping array that stores the global 2D position of each grid cell, each processor manages the range from its begg to endg
`procinfo`	Stores information on the number of clumps, clump identifiers, size of grid cells, and the start and end indices of subgrid elements within the corresponding MPI process.
`bounds_type`	Manages data related to the total number of subgrid components in grid cells, including their starting and ending indices within either clumps or MPI processes.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97626-1_12

4 User-Defined Decomposition Framework

4.1 Framework Description

As mentioned hereinbefore, in the default ELM decomposition framework, domain partitioning relies on a static round-robin distribution of land grid cells across processors to maintain load balance. However, this approach lacks flexibility in defining custom partitioning strategies, potentially leading to inefficiencies in computation, communication, and I/O at the kilometer scale.

To address this limitation, we introduce a user-defined partition framework that provides a flexible mechanism, allowing users to modify the `amask` values (user-provided decomposition) in the domain file to control domain decomposition. The `amask` is a 1D array where each grid cell is assigned a value, indicating whether it is a land unit (≥ 1 denotes land, and 0 indicates ocean) and defining the affinity or clump assignment of each grid cell (ranging from 1 to `nclumps`). This method directly maps `amask` to `lcid`, which records the clump assignment for each grid cell (see Table 1). This mapping enables fine-grained control over decomposition, allowing users to apply various partitioning strategies without modifying the core algorithm in ELM. Algorithm 1 illustrates the process of iterating through each grid cell, reading `amask` values, and assigning them to `lcid` to ensure proper workload distribution. Here, `lns` represents the total number of grid cells, while `numg`, `lcid`, `procinfo`, and `clumps` are utilized later when updating `gcd2glo`.

The implementation of this framework requires modifications to ELM's domain decomposition logic. A key modification involves adapting the `gdc2glo` mapping, which converts between 1D and 2D grid structures. Rather than relying on predefined round-robin assignments, the framework employs `lcid` to determine the number of land units assigned to each clump of each processor.

Algorithm 1: User-Defined Domain Decomposition in Fortran.

```
for ln = 1 to lns do
    // Loop over each grid cell; lns represents the 1D domain size
    if amask(ln) > 0 then
        // Check if the current grid cell is a land unit
        numg = numg + 1 // Count active land grid cells
        lcid(ln) = amask(ln) // Assign processor ID from mask (amask is
            the mask value read from the domain file)
        if iam == clumps(lcid(ln))%owner then
            procinfo%ncells = procinfo%ncells + 1 // Update processor
                cell count
        end
        clumps(lcid(ln))%ncells = clumps(lcid(ln))%ncells + 1 // Update
            clump cell count
    end
end
```

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97626-1_12

This method ensures precise updates of `begg` and `endg`, which define the computational range (refer to Table 1). In detail, `lcid` provides the 1D position of each assigned land unit, facilitating efficient indexing. These 1D indices are mapped back to their 2D global positions (i.e., latitude `lni` and longitude `lnj`), ensuring that `gdc2glo` is correctly updated while preserving spatial relationships.

Another complexity arises from ELM's hierarchical grid structure, where each primary land grid cell contains six subgrid levels. A different partition necessitates corresponding updates to the `bounds_type` structures (see Table 1) at each subgrid level, which governs memory allocation and communication. To ensure consistency across different configurations, rigorous validation is also performed.

4.2 Partition Strategies for Demonstration

Different partitioning strategies impact computation, communication, and I/O performance differently. Therefore, we select six representative partitioning strategies across three categories: (i) static, predefined partitioning, (ii) adaptive, data-driven methods, and (iii) watershed-based strategy. These strategies represent different workload distributions and data access patterns, as visualized in Fig. 1, where a 10×10 grid is distributed across 10 clumps.

Static, predefined partitioning. 1D Partition divides the computational domain into contiguous segments, ensuring that each processor is assigned a continuous block of grid cells. Regular round robin distributes grid cells sequentially across all processors in a cyclic manner. Block round robin provides an intermediate approach between these two, offering a comparative perspective. It assigns

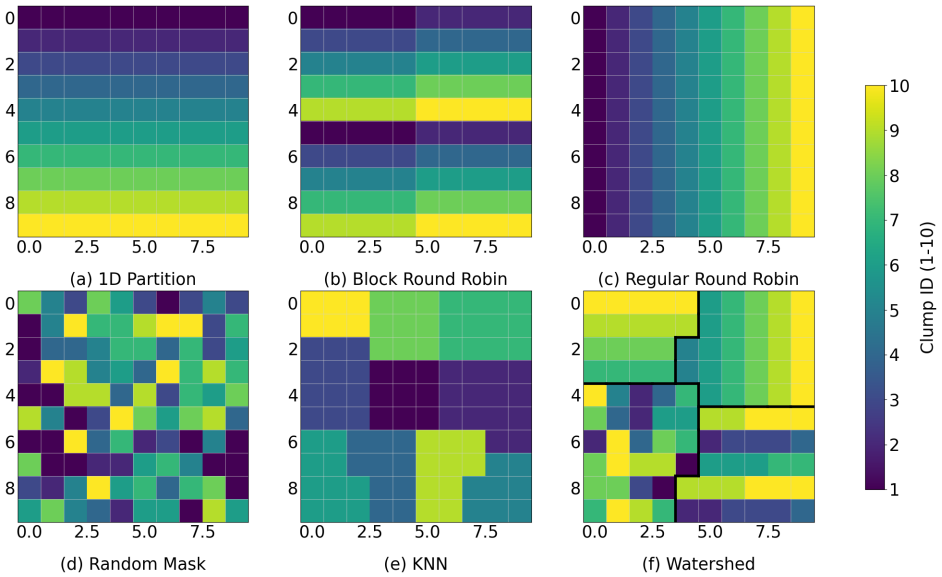


Fig. 1: Visualization of various partitioning strategies using ten clumps.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97626-1_12

gridcells in fixed-size blocks, cyclically distributing them among processors. The block size (e.g., 5 in Fig. 1(b)) determines how many consecutive grid cells are assigned to a single processor before moving to the next.

Adaptive, data-driven partitioning. Random Mask allows users to directly assign processors to specific grid cells by modifying the land mask randomly, presenting an extreme case in partitioning strategies. K-Nearest Neighbors (KNN) employs a clustering algorithm based on spatial proximity or feature similarity to group neighboring gridcells together.

Watershed-based partitioning. Watershed-based partitioning divides the computational domain based on the demographic partitioning of the Tennessee Valley Authority (TVA) region (details in Section 5.2). Within these subdomains, the independent partitioning strategies mentioned above are applied to assess effectiveness in grouping grid cells based on hydrological characteristics.

5 Performance Results and Analysis

5.1 Experimental Settings

The experiments are conducted on two high-performance computing systems from the Oak Ridge Leadership Computing Facility (OLCF):

- `Baseline`: A CPU-based system composed of 180 compute nodes. Each node is equipped with two AMD EPYC 7713 processors, providing a total of 128 cores running at 2.0 GHz. The nodes are configured with either 256 GB or 512 GB of memory. It offers 2.3 PB of shared storage through the Wolf2 GPFS filesystem, making it suitable for large-scale simulations with high I/O throughput requirements.
- `Frontier`: A GPU-accelerated supercomputer utilizing AMD hardware, comprising 9,408 compute nodes. Each node integrates a 64-core AMD Optimized 3rd Gen EPYC processor alongside four AMD MI250X GPUs, supported by 512 GB of DDR4 memory. The system is connected to Orion, a parallel filesystem based on Lustre and HPE ClusterStor, featuring a 679 PB usable namespace.

In this study, the MPI binding policy is set to by-core, meaning that each core hosts exactly one process. Without loss of generality, each process includes one clump for simplicity. The block size remains fixed for block round-robin in each application throughout the experiments. Additionally, under the KNN strategy, the number of assigned blocks can be adjusted based on the number of processes. This work utilizes ELM in land-only mode, as detailed in Section 3.

5.2 Application Settings

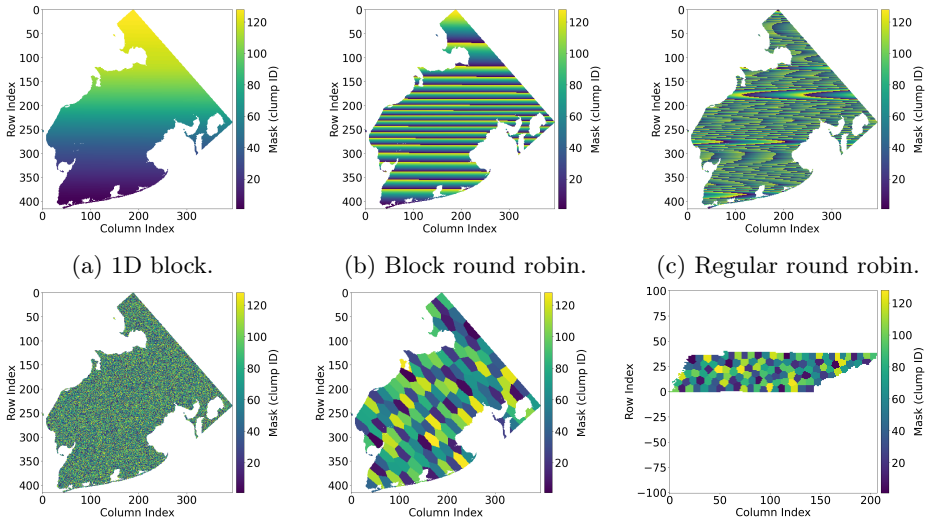
To evaluate different partition strategies, we select four distinct regions: AKSP, AKSPx10, TES_TVA, and TN. These domains allow us to assess the impact of different partition methods.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97626-1_12

– **AKSP**: The AKSP domain covers the Seward Peninsula in western Alaska, spanning approximately 330 km in length and 145–225 km in width, with a total area of 72,083 km^2. The simulation employs a high-resolution 1 km × 1 km grid, resulting in 72,083 grid cells.
– **AKSPx10**: This experiment scales up the AKSP domain to evaluate partitioning methods in ELM. The AKSPx10 simulation maintains the same spatial characteristics but expands to 720,830 land grid cells with corresponding subgrid components. Its presentation is identical to that of the AKSP domain.
– **TES_TVA**: Designed to test the Watershed-Based Partition method, this domain is first subdivided into four distinct subdomains within the Tennessee Valley Authority (TVA)—Mississippi (MS), Ohio (OH), Tennessee (TN), and Gulf (GULF)—each representing a hydrologically significant region. The domain consists of 10,357 grid cells, reflecting natural watershed boundaries.
– **TN**: This focuses on the TN region within the TVA watershed, encompassing 6,317 grid cells and capturing its hydrological and ecological characteristics.

5.3 Decomposition Map

Fig. 2 presents the partition map of different decomposition strategies (see Fig. 1) in real applications (AKSP and TN) generated on `Baseline` with 128 processes. Grid cells in E3SM are stored in a structured 1D format but are mapped to a 2D computational grid using row and column indices. The column index corresponds to the x-axis direction (longitude), while the row index represents the y-axis di-



(a) 1D block. (b) Block round robin. (c) Regular round robin.

(d) Random mask. (e) KNN in AKSP. (f) KNN in TN.

Fig. 2: Decomposition map of AKSP ((a)-(e)) and TN ((f)).

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97626-1_12

rection (latitude). These indices are derived by normalizing spatial coordinates relative to their minimum values and dividing by the grid resolution. In Fig. 2, grid cells are ordered (1) starting from the left corner for AKSK in Fig. 2 (a)-(e) (i.e., (column index, row index): (0, largest_value_in_y-axis)) and (2) starting from the left center for TN in Fig. 2 (f) (i.e., (column index, row index): (0, 0)), which are then progressed row by row. AKSP is based on an older mapping system, so its actual starting point does not align with modern geographic conventions, resulting in a rotated and flipped representation. From these figures, we observe slight differences in the decomposition results compared to Figure 1, primarily due to irregularities in the domain file and the skipping of ocean grid cells (see Algorithm 1). These results illustrate the decomposition of real-world applications under different strategies, emphasizing the complexities introduced by domain-specific constraints.

5.4 Impact of Partitioning Methods on Computation

Table 2 presents a comparison of computational time across different partitioning strategies for various test cases on `Baseline`. Each case type is evaluated under different node configurations. The computational time is determined by subtracting lnd component I/O time from the lnd component total running time. While the similarity in computational times across different partitioning methods within each case suggests effective load balancing, slight variations are observed. Static, predefined partitioning methods (1D partition, block round robin, and regular round robin) distribute grid cells evenly among processors, resulting in better workload balance and reduced computational time. In contrast, adaptive, data-driven partitioning methods (random mask and KNN) exhibit slightly higher computational times due to variations in workload distribution. For instance, Random Mask may lead to certain processors receiving more grid cells than others, causing computational load imbalance.

The results also demonstrate both weak and strong scalability. Consider the 1D Partition as an example. Weak scalability is assessed by comparing AKSP on a single node (337.02s) with AKSPx10 on 10 nodes (432.12s), showing that as problem size increases proportionally with computational resources, performance remains stable. Strong scalability is evident in the AKSP cases, where compu-

Table 2: Comparison of Computational Time (Seconds). Red: worst computational time; Blue: best computational time.

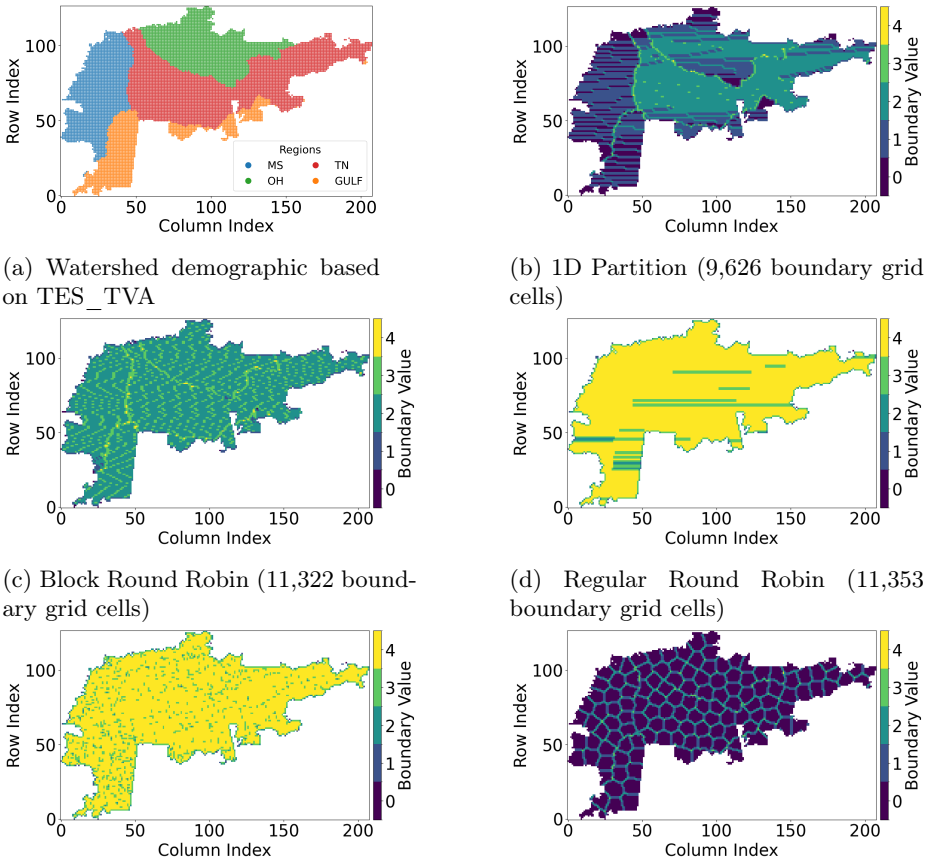
Case Type	1D Partition	Block Round Robin	Round Robin	Random Mask	KNN
AKSP 1 node	**337.02**	338.97	338.44	343.16	**346.59**
AKSP 5 nodes	**63.14**	64.24	64.14	**62.02**	62.99
AKSP 10 nodes	**31.74**	33.73	33.78	**35.11**	34.67
AKSPx10 10 nodes	**432.12**	**521.32**	491.77	511.44	516.67
TVA Watershed 1 node	68.17	63.44	**63.09**	**70.82**	64.90
TN 1 node	34.61	33.34	32.44	**36.55**	**32.29**

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97626-1_12

tational time decreases from 337.02s (1 node) to 63.14s (5 nodes) to 31.74s (10 nodes), indicating improved efficiency with increasing computational resources.

5.5 Impact of Partitioning Methods on Communication

Fig. 3 (a) visualizes the four distinct subdomains within TVA, and Fig. 3 (b)–(f) presents the boundary grid cell heatmap for different partitioning strategies applied to the TVA watershed sub-domain on `Baseline` with 128 processes. The partitioning method for each sub-domain remains consistent in each setting, while each sub-domain can adopt a different decomposition. Each partitioning method results in a unique distribution of boundary grid cells, which are computed using an adjacency-based method, indicating the extent of inter-process



(a) Watershed demographic based on TES_TVA

(b) 1D Partition (9,626 boundary grid cells)

(c) Block Round Robin (11,322 boundary grid cells)

(d) Regular Round Robin (11,353 boundary grid cells)

(e) Random mask (11,351 boundary grid cells)

(f) K-Nearest Neighbors (3,374 boundary grid cells)

Fig. 3: Different partition strategies applied to watershed sub-domain

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97626-1_12

communication required under each strategy. For each grid cell, its process/-clump ID is compared with the four neighboring grid cells (top, bottom, left, and right). If the ID of any neighboring grid cell differs, the boundary count for that grid cell is incremented. Consequently, the boundary value of each grid cell ranges from 0 to 4, indicating the number of communications if lateral flow is enabled. The total number of boundary grid cells is obtained by counting all grid cells where this computed value is greater than zero. From the results shown in the figures and the boundary grid cell counts, KNN exhibits the lowest number of boundary grid cells, thereby introducing the lowest communication overhead when lateral connectivity is activated. The 1D partition performs worse than KNN due to irregularities in the domain file and the omission of ocean grid cells, as discussed in Section 5.3.

5.6 Impact of Partitioning Methods on I/O Performance

Table 3 compares the effect of different partitioning strategies on I/O, including five settings on `Baseline` and two on `Frontier`. The table reports the total running time (TOT Time) and land component running time (LND Time), where TOT Time includes the time of LND, CPL (coupler), and ATM (atmosphere) [3]. Km-ELM is deployed in land-only mode (details in Section 3), so LND contributes to most of the time-to-solution. The computational time across

Table 3: Impacts of Partitioning Methods on I/O (Seconds). Red: worst running time; Blue: best running time.

Time (s)	1D Partition	Block Round Robin	Regular Round Robin	Random Mask	KNN
AKSP 1 node (Baseline)					
TOT Time	387.76	413.83	636.52	1277.80	435.00
LND Time	375.61	393.17	613.96	1263.57	415.85
AKSP 2 node (Frontier)					
TOT Time	364.25	374.16	382.01	390.27	387.72
LND Time	359.15	370.08	363.02	368.02	370.42
AKSP 10 nodes (Baseline)					
TOT Time	93.73	99.59	96.58	113.24	102.32
LND Time	86.71	85.88	83.41	97.61	87.34
AKSPx10 10 nodes (Baseline)					
TOT Time	901.98	915.99	1191.92	1168.17	1171.88
LND Time	866.31	840.01	1060.50	1106.21	1081.51
AKSPx10 20 nodes (Frontier)					
TOT Time	401.23	420.74	432.02	432.03	432.02
LND Time	367.64	379.09	385.38	389.24	385.38
TVA 1 node (Baseline)					
TOT Time	74.54	77.74	82.02	79.74	83.72
LND Time	70.54	72.89	72.62	72.58	78.09
TVA Watershed 1 node (Baseline)					
TOT Time	82.28	75.98	77.61	87.57	81.33
LND Time	76.30	70.85	70.93	79.32	74.10
TN 1 node (Baseline)					
TOT Time	42.02	41.48	41.50	53.45	46.10
LND Time	39.90	38.30	37.29	45.11	39.73

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97626-1_12

different partitioning strategies remains nearly identical (see Section 5.4), and lateral flow is under development in km-ELM, so we use the TOT/LND time as the I/O effects. Additionally, each `Baseline` node consists of 128 cores, while each node on `Frontier` has 64 cores. To maintain consistency in computational resources, all experiments on `Frontier` are conducted using twice the number of nodes as in the `Baseline` setup. Km-ELM optimizes I/O efficiency by employing the *SCORPIO* parallel I/O library [20, 21], which organizes distributed data before transferring it to underlying storage libraries such as *PnetCDF* and *NetCDF*. These libraries improve write performance by grouping MPI processes into aggregators, which manage collective data output to parallel file systems like *Lustre* [22] and *GPFS* [23]. The aggregation process reduces file fragmentation and enhances overall throughput.

From this table, partitioning methods significantly impact I/O performance due to differences in data arrangement, where 1D partition usually is the best and random mask the worst. When using a 1D partition strategy, the grid cells are already assigned to processors in a contiguous and sequential manner. As a result, *SCORPIO* can efficiently process the data with minimal rearrangement, reducing I/O overhead. In contrast, methods like random mask introduce non-contiguous data distribution, increasing the time required for *SCORPIO* to restructure the output before writing it to storage. Given the absence of lateral flow in the current experiments, the reduced data reordering time makes 1D partition the most efficient method in most scenarios. Furthermore, comparing the results of AKSPx10 between the `Baseline` and `Frontier` (901.98s on `Baseline` and 401.23s on `Frontier` for 1D partition), `Frontier` demonstrates shorter I/O time (515.41s on `Baseline` and 29.66s on `Frontier` for 1D partition), indicating the improved filesystem on `Frontier`.

6 Conclusion and Future Work

This study introduces FUDD, a flexible framework for user-defined domain decomposition, and explores its application within km-scale ELM. The analysis examines six distinct partitioning strategies, categorized into three main types: static predefined approaches, adaptive data-driven techniques, and watershed-based methods, each representing different workload distributions and data access behaviors. By applying FUDD in real-world scenarios across two clusters, we evaluate FUDD's impact on computation, communication, and I/O. The results provide a foundation for advancing scalable domain decomposition strategies, contributing to the development of efficient next-generation Earth system modelings. Looking ahead, we plan to extend the capabilities of FUDD by integrating it with the GPU-enabled version of ELM, known as uELM [5], to further explore its effectiveness in heterogeneous environments. Additionally, we intend to evaluate FUDD when the lateral flow in km-ELM becomes available. Furthermore, we aim to develop an automated mechanism for intelligently selecting the most suitable partitioning strategy based on the characteristics of a given application.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97626-1_12

Acknowledgments. This research is supported as part of the Energy Exascale Earth System Model (E3SM) project funded by DOE's Office of Science, Office of Biological and Environmental Research. This project leverages the Daymet data product funded by NASA and uses resources of the Oak Ridge Leadership Computing Facility, supported by DOE's Office of Science under Contract No. DE-AC05-00OR22725.

References

1. DOE. https://e3sm.org/, 2024.
2. Fengming Yuan, Dali Wang, Shih-Chieh Kao, Michele Thornton, Daniel Ricciuto, Verity Salmon, Colleen Iversen, Peter Schwartz, and Peter Thornton. An ultrahigh-resolution e3sm land model simulation framework and its first application to the seward peninsula in alaska. *Journal of Computational Science*, 73:102145, 2023.
3. Dali Wang, Chen Wang, Qinglei Cao, Peter Schwartz, Fengming Yuan, Jayesh Krishna, Danqing Wu, Danial Ricciuto, Peter Thornton, Shih-Chieh Kao, et al. Kilometer-scale e3sm land model simulation over north america. *arXiv preprint arXiv:2501.11141*, 2025.
4. Forrest M Hoffman, Mariana Vertenstein, Hideyuki Kitabata, and James B White III. Vectorizing the community land model. *The International Journal of High Performance Computing Applications*, 19(3):247–260, 2005.
5. Peter D. Schwartz, Dali Wang, Fengming Yuan, and Peter E. Thornton. Developing ultrahigh-resolution e3sm land model for gpu systems. In *Computational Science and Its Applications – ICCSA 2023*, volume 13956, pages 277–290, Cham, Switzerland, June 2023. Springer.
6. SM Burrows, M Maltrud, X Yang, Q Zhu, N Jeffery, X Shi, D Ricciuto, S Wang, G Bisht, J Tang, et al. The doe e3sm v1. 1 biogeochemistry configuration: Description and simulated ecosystem-climate responses to historical changes in forcing. *Journal of Advances in Modeling Earth Systems*, 12(9):e2019MS001766, 2020.
7. Yang Xu, Dali Wang, Tomislav Janjusic, Wei Wu, Yu Pei, and Zhuo Yao. A web-based visual analytic framework for understanding large-scale environmental models: A use case for the community land model. *Procedia Computer Science*, 108:1731–1740, 2017.
8. W. Zheng, D. Wang, and F. Song. Xscan: An integrated tool for understanding open source community-based scientific code. In *International Conference on Computational Science*, pages 226–237. Springer, 2019.
9. Dali Wang, Peter Schwartz, Fengming Yuan, Peter Thornton, and Weijian Zheng. Towards ultra-high-resolution e3sm land modeling on exascale computers. *Computing in Science & Engineering*, (01):1–14, 2022.
10. Dalei Hao, Gautam Bisht, Meng Huang, Po-Lun Ma, Teklu Tesfa, Wei-Liang Lee, Yu Gu, and L Ruby Leung. Impacts of sub-grid topographic representations on surface energy balance and boundary conditions in the e3sm land model: A case study in sierra nevada. *Journal of Advances in Modeling Earth Systems*, 14(4):e2021MS002862, 2022.
11. Peter E Thornton, Michele M Thornton, Benjamin W Mayer, Nate Wilhelmi, Yaxing Wei, Ranjeet Devarakonda, and Robert B Cook. Daymet: Daily surface weather data on a 1-km grid for north america, version 2. Technical report, Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States), 2014.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97626-1_12

12. Shih-Chieh Kao, Michele Thornton, Peter E Thornton, and Rupesh Shrestha. Gridded sub-daily climate forcings for north america based on daymet and gswp3 (daymet-gswp3). Technical report, Oak Ridge National Laboratory (ORNL), Oak Ridge, TN (United States). Oak . . . , 2024.
13. Q. Han, Y. Zeng, L. Zhang, C. Wang, E. Prikaziuk, Z. Niu, and B. Su. Global long term daily 1 km surface soil moisture dataset with physics informed machine learning. *Scientific Data*, 10(1):101, 2023.
14. A. S. Donahue, P. M. Caldwell, L. Bertagna, H. Beydoun, P. A. Bogenschutz, A. M. Bradley, T. C. Clevenger, J. Foucar, C. Golaz, O. Guba, W. Hannah, B. R. Hillman, J. N. Johnson, N. Keen, W. Lin, B. Singh, S. Sreepathi, M. A. Taylor, J. Tian, C. R. Terai, P. A. Ullrich, X. Yuan, and Y. Zhang. To exascale and beyond—the simple cloud-resolving e3sm atmosphere model (scream), a performance portable global atmosphere model for cloud-resolving scales. *Journal of Advances in Modeling Earth Systems*, 16(7):e2024MS004314, 2024. e2024MS004314 2024MS004314.
15. Omega- future e3sm ocean model. `https://climatemodeling.science.energy.gov/news/omega-future-e3sm-ocean-model`. Accessed: 2010-09-30.
16. Nicolas Viovy. Cruncep version 7-atmospheric forcing data for the community land model. *Research Data Archive at the National Center for Atmospheric Research, Computational and Information Systems Laboratory*, 10, 2018.
17. Taotao Qian, Aiguo Dai, Kevin E Trenberth, and Keith W Oleson. Simulation of global land surface conditions from 1948 to 2004. part i: Forcing data and evaluations. *Journal of Hydrometeorology*, 7(5):953–975, 2006.
18. Kei Yoshimura and Masao Kanamitsu. Incremental correction for the dynamical downscaling of ensemble mean atmospheric fields. *Monthly weather review*, 141(9):3087–3101, 2013.
19. Peter E Thornton, Rupesh Shrestha, Michele Thornton, Shih-Chieh Kao, Yaxing Wei, and Bruce E Wilson. Gridded daily weather data for north america with comprehensive uncertainty quantification. *Scientific Data*, 8(1):1–17, 2021.
20. Jayesh Krishna and Danqing Wu. Software for caching output and reads for parallel i/o. `https://github.com/E3SM-Project/scorpio`, 2020.
21. Edward Hartnett and Jim Edwards. The parallelio (pio) c/fortran libraries for scalable hpc performance. 01 2021.
22. SUN. High-performance storage architecture and scalable cluster file system. Tech. rep., Sun Microsystems, Inc, 2007.
23. Frank Schmuck and Roger Haskin. Gpfs: A shared-disk file system for large computing clusters. In *Proceedings of the 1st USENIX Conference on File and Storage Technologies*, FAST '02, page 19–es, USA, 2002. USENIX Association.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97626-1_12