Parallel Distributed Hydrology Soil Vegetation Model (DHSVM) Using Global Arrays

William Perkins[1,1], Zhuoran Duan[1,1], Ning Sun[1,1], Mark Wigmosta[2,2], Marshall Richmond[1,1], Xiaodong Chen[1,1], and L. Ruby Leung[1,1]

[1]Pacific Northwest National Laboratory
[2]University of Washington,Pacific Northwest National Laboratory

November 30, 2022

Abstract

The Distributed Hydrology Soil Vegetation Model (DHSVM) code was parallelized for distributed memory computers using the Global Arrays (GA) programming model. To analyze parallel performance, DHSVM was used to simulate the hydrology in two river basins of significant size located in the northwest continental United States and southwest Canada at 90˜m resolution: the (1) Clearwater (25,000˜km) and (2) Columbia (668,000˜km) River basins. Meteorological forcing applied to both basins was dynamically down-scaled from a regional reanalysis using the Weather Research and Forecasting (WRF) model and read into DHSVM as 2D maps for each time step. Parallel code speedup was significant. Run times for 1-year simulations were reduced by an order of magnitude for both test basins. A maximum parallel speedup of 105 was attained with 480 processors while simulating the Columbia River basin. Speedup was limited by input-dominated tasks, particularly the input of meteorological forcing data.

Parallel Distributed Hydrology Soil Vegetation Model (DHSVM) Using Global Arrays

William A. Perkins[a,*], Zhuoran Duan[a], Ning Sun[a], Mark S. Wigmosta[a], Marshall C. Richmond[a], Xiaodong Chen[b], L. Ruby Leung[b]

[a]*Hydrology Group, Pacific Northwest National Laboratory, Richland, WA, USA*
[b]*Atmospheric Sciences and Global Change, Pacific Northwest National Laboratory, Richland, WA, USA*

Abstract

The Distributed Hydrology Soil Vegetation Model (DHSVM) code was parallelized for distributed memory computers using the Global Arrays (GA) programming model. To analyze parallel performance, DHSVM was used to simulate the hydrology in two river basins of significant size located in the northwest continental United States and southwest Canada at 90 m resolution: the (1) Clearwater (25,000 km^2) and (2) Columbia (668,000 km^2) River basins. Meteorological forcing applied to both basins was dynamically down-scaled from a regional reanalysis using the Weather Research and Forecasting (WRF) model and read into DHSVM as 2D maps for each time step.

Parallel code speedup was significant. Run times for 1-year simulations were reduced by an order of magnitude for both test basins. A maximum parallel speedup of 105 was attained with 480 processors while simulating the Columbia River basin. Speedup was limited by input-dominated tasks, particularly the input of meteorological forcing data.

Keywords: watershed hydrology, distributed hydrology model, high-performance computing, parallel computing, Columbia River basin, Clearwater River basin

Highlights

- The well established Distributed Hydrology Vegetation Soil Model (DHSVM) was parallelized for distributed memory platforms.
- The Global Arrays (GA) partitioned global address space (PGAS) library for distributed arrays was used for inter-process communication.
- Parallel DHSVM was used to simulate the hydrology of two large river basins, with areas of 25,000 and 668,000 square kilometers, at a 90 meter resolution.
- Maximum parallel speed up of 105 was measured using 480 processors with the Columbia River basin simulation.

*Corresponding author. Pacific Northwest National Laboratory, P.O. Box 999 MSIN K9-36, Richland, WA, USA, 99354. *E-mail address*: william.perkins@pnnl.gov

Software Availability

Program Title: Parallel DHSVM
Description: Distributed Hydrology Vegetation Soil Model (DHSVM)
Platform: Linux, Mac OS X
Source Language: C, C++
Cost: Free
License: public domain
Availability: Source code available on Github (`https://github.com/pnnl/DHSVM-PNNL`), "`parallel`" branch

1. Introduction

The Distributed Hydrology Vegetation Soil Model (DHSVM, Wigmosta et al., 1994) is a spatially distributed, physics-based hydrology model that simulates the overland and subsurface hydrological processes influenced by climate, topography, soil, and vegetation. DHSVM is composed of a two-layer canopy model, an energy-balance two-layer snow model, a multi-layer soil model, and three-dimensional surface and subsurface flow-routing models. These models allow for characterization of hydrological processes including canopy and topographic shading, canopy interception, evapotranspiration, snow accumulation and melt, and water movement overland and through the soil to streams and rivers. In an extensive review of 30 hydrological models (Beckers et al., 2009), DHSVM was identified to be best suited for modeling mountain hydrology in forested environments.

Initially developed in the early 1990s (Wigmosta et al., 1994), DHSVM has been applied extensively, particularly in forested, mountainous, snowfall-dominated regions, to characterize the hydrologic regime and project potential changes with changing climate and landscape (Storck et al., 1998; Storck and Lettenmaier, 1999; Leung and Wigmosta, 1999; Thyer et al., 2004; Cuo et al., 2009; Cristea et al., 2014; Livneh et al., 2015; Cao et al., 2016; Sun et al., 2018). Subsequent adaptations have extended the capability of DHSVM to represent urban landscapes with impervious surfaces and runoff detention (Cuo et al., 2008), glacio-hydrological dynamics (Naz et al., 2014; Frans et al., 2015, 2018), river thermal dynamics (Sun et al., 2015; Cao et al., 2016), urban water quality (Sun et al., 2016), and forest-snow interactions in canopy gaps (Sun et al., 2018).

With the increasing availability of high-resolution satellite products, e.g., Light Detecting and Ranging and advances in high-performance computing systems and data storage, there is evolving interest in exploring hydrologic fluxes and state variables at progressively higher spatial resolutions for applications ranging from regional to global scales (Lettenmaier et al., 2015). High-resolution, spatially distributed modeling capabilities are particularly important for representing complex mountain hydrology that is highly affected by heterogeneous terrain and strong climate gradients with elevation. A spatially lumped modeling approach with sparsely distributed observation networks can limit our ability to understand and predict the implications of changing climate and landscape on available water for extreme runoff events, regional water supplies,

and associated reservoir operations for hydropower and other water allocations (Bales et al., 2006).

While DHSVM has been under constant development since its inception, it has always been a serial code. Its computational performance has been tied to the performance of a single processor. Parallelization is a good strategy for helping meet these and future simulations needs. A number of examples in the literature describe parallel hydrological models. The majority (e.g., Hwang et al., 2014; Liu et al., 2014, 2016; Adriance et al., 2019) seem to favor small shared memory platforms using OpenMP (Dagum and Menon, 1998). A few (Vivoni et al., 2011; Kumar and Duffy, 2016) target distributed memory systems using the Message Passing Interface (MPI; MPI Forum, 2018).

In this work, DHSVM was made into a parallel code while maintaining most of its existing capability. The parallel code development was aimed at large distributed memory clusters, but portability to smaller multiprocessor, shared memory systems, such as desktops and laptops, was maintained. An alternate interprocess communication programming model, Global Arrays (GA, Nieplocha et al., 2006; Manojkumar et al., 2012) was used. GA provides a partitioned global address space (PGAS) and implements one-sided communication protocols.

To demonstrate its utility and scalability, the parallel DHSVM was used to simulate runoff from two basins of significant size. This work is limited to demonstrating the performance of parallel DHSVM.

2. Methods

2.1. Hydrologic Process Representation

The DHSVM domain is divided into an array of rectangular cells (Figure 1). Cell size is determined by the resolution of the Digital Elevation Model (DEM) used. A mask is used to denote which cells in the domain are active, typically encompassing a watershed that drains to a single point. Within each cell, a water mass and energy balance is maintained. Excess surface water is routed down slope overland; excess drainage to the subsoil layer is routed downgradient to neighboring cells until reaching the channel network.

As much as possible, DHSVM uses physically based representations to compute the movement of water and energy through the domain. The details of DHSVM hydrologic process representation are presented elsewhere (e.g., Wigmosta et al., 1994; Wigmosta and Lettenmaier, 1999; Wigmosta et al., 2002; Cuo et al., 2009; Naz et al., 2014; Frans et al., 2018; Sun et al., 2018). Brief descriptions of some processes important to code parallelization are presented here.

2.1.1. Cell Energy/Water Balance

A DHSVM cell consists of a set of soil layers, a set of snowpack layers (when present) and a multi-level vegetation canopy. Meteorological forcing data are used to drive the energy balances in the snowpack, resulting in melt and/or accumulation, and in the vegetation canopy, resulting in evapotranspiration.

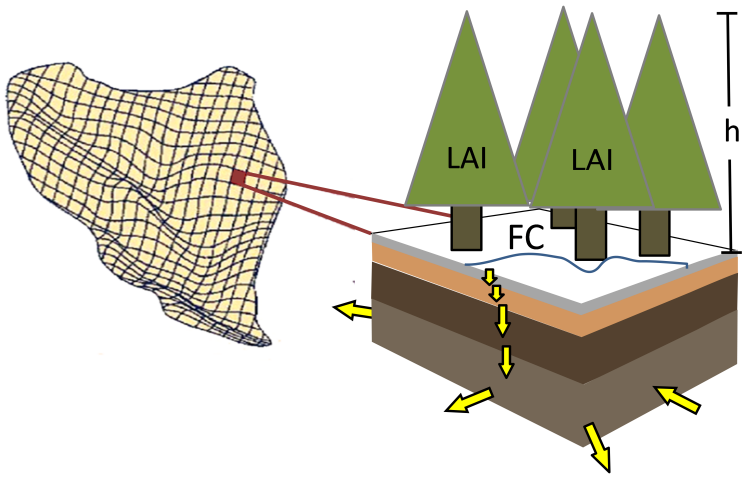


Figure 1: Schematic representation of water movement in the DHSVM domain. The DHSVM domain is divided into rectangular cells in which water and energy balance is maintained. Excess surface and subsurface water is routed to a channel network. LAI is leaf area index, FC stands for fractional cover of forest canopy, and h is canopy height.

Movement of water in the cell's soil layers is simulated. This includes infiltration or exfiltration, evaporation from the soil surface, evapotranspiration from soil layers in which vegetation has roots, vertical saturated and unsaturated water movement between layers, and drainage to a subsurface soil layer.

All of these calculations take place within the cell, independent of its neighbors. The results are volumes of water in each soil and snow layer and on the surface.

2.1.2. Surface and Subsurface Routing

The surface and subsurface volumes computed in the cell energy/water balance are routed to neighboring cells. The DHSVM routing schemes are documented by Wigmosta et al. (1994), Wigmosta and Lettenmaier (1999), and Wigmosta et al. (2002). Both surface and subsurface routing work with a similar algorithm.

A gradient, based on the ground surface or water (table) surface, is used to determine the direction and magnitude of flow for each cell. In a cell, discharge to each neighboring cell is computed and stored. Surface water flux from active cell ij to its kth down slope neighbor is computed as

$$q_{o_{ij_k}} = w_{ij_k} v_{ij_k} y_{ij} \tag{1}$$

where w_{ij_k} is the flow width in the k direction, v_{ij_k} is the overland flow velocity, and y_{ij} is the overland flow depth. Subsurface flow from active cell ij to its kth downgradient neighbor is computed as

$$q_{s_{ij_k}} = w_{ij_k} \beta_{ij_k} T_{ij}(z, D) \tag{2}$$

where β_{ij_k} is the cell water table or land slope and $T_{ij}(z, D)$ is the soil transmissivity,

computed as

$$T_{ij}(z, D) = \frac{K_{ij}}{f_{ij}} \left(e^{-f_{ij} z_{ij}} - e^{-f_{ij} D_{ij}} \right)$$

where K_{ij} is the cell lateral saturated hydraulic conductivity, z_{ij} is the depth to the water table, f_{ij} is a decay coefficient, and D_{ij} is the cell soil thickness.

These fluxes are computed for every cell in the domain. Each cell accumulates the inflow from its upgradient neighbors, discharges to adjacent down gradient cells, and adjusts surface and subsurface volumes accordingly.

2.1.3. Stream Channel Network

DHSVM uses a stream channel network to route excess surface water and intercepted subsurface flow to the watershed outlet. The stream channel network is represented by a cascade of linear reservoirs (Wigmosta et al., 2002).

After surface and subsurface routing is complete, computed stream channel interception of surface and subsurface flow is accumulated for each cell in which a stream channel lies. The intercepted water volume is summed and used as lateral inflow for each stream segment. The lateral inflow is then routed through the network.

The outflow rate of segment i at time $t + 1$ is given by

$$O_i^{t+1} = \left(I_i^{t+1} + L_i^{t+1} \right) - \left(S_i^{t+1} - S_i^t \right) \frac{1}{\Delta t} \tag{3}$$

where I_i^t is the inflow rate at time t to segment i from upstream segment(s), L_i^t is the lateral inflow at time t into segment i, Δt is the time step between t and $t + 1$, and S_i^t is the segment storage at time t, computed using

$$S_i^{t+1} = \frac{1}{K} \left(I_i^{t+1} + L_i^{t+1} \right) + X \left[S_i^t - \frac{1}{K} \left(I_i^{t+1} + L_i^{t+1} \right) \right] \tag{4}$$

in which

$$K = \frac{\sqrt{S_o} R^{\frac{2}{3}}}{nl}$$

and

$$X = e^{-K \Delta t}$$

where S_o is the channel slope, n is Manning's coefficient, l is the channel length, R is the hydraulic radius, which is assumed to be a constant 75% of the bank height, and Δt is the time step.

2.2. Code Parallelization

The multiple instruction, multiple data (MIMD, Wilkinson and Allen, 1998) parallel model was used. This approach targets large, distributed memory systems (i.e. clusters), but the approach should work fine for smaller, shared memory systems (multiprocessor desktops and laptops) without modification. In the MIMD model, each processor is assigned its own data to work on independently and some communication

layer is required to exchange data between processors when needed. In this case, each DHSVM process is assigned a non-overlapping rectangular subset of the active cells in domain.

The goal was to make DHSVM as fast as practical while retaining as much of its existing behavior as possible. DHSVM is a relatively large and complicated code. Resources were not available to design and code a parallel DHSVM from the ground up. This in some ways limited the parallelization approach and results.

2.2.1. Interprocess Communication

Inter-process communication in DHSVM was implemented through the use of GA (Nieplocha et al., 2006; Manojkumar et al., 2012). GA is a "partitioned global address space library for distributed arrays". GA provides a distributed, random access, multi-dimensional array data structure. Such an array is consistent with the internal DHSVM data structures, so most of the serial code structure could be retained. In addition, nearly all of the required interprocess communication consists of floating point values, which simplifies coding.

In general, DHSVM interprocess communication is all cell-based numeric values (i.e., rectangular arrays). In a typical communication scenario, a GA structure is created. Transfers of values are made from local memory to the GA (put) and from the GA to local memory (get). Other operations are available, like "accumulate" where values in local memory are summed into the GA.

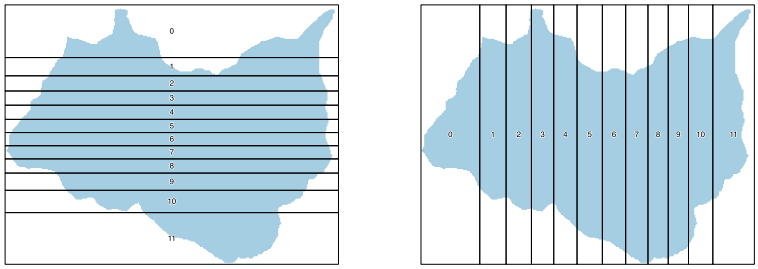
GA can use several underlying communication protocols, depending on the underlying hardware. The most commonly used are based on MPI and can be used on almost any platform that supports MPI. These range from large clusters to laptops—any shared or distributed memory system for which MPI is available (Dinan et al., 2012). DHSVM relies entirely on the GA application programming interface (API). There are no direct calls to any other parallel communication interface.

2.2.2. Domain Decomposition

The most straightforward approach to parallelization was to distribute cell-based calculations across processors. A divide and conquer strategy was implemented that has some similarity to the strategy used by Hwang et al. (2014). Each process was assigned a non-overlapping rectangular region of the original domain. As shown in Figure 2, the region assigned to a process may be a collection of rows (`STRIPEY`) or a collection of columns (`STRIPEX`).

An algorithm similar to Simeone's (1986) is used to evenly distribute the *active* cells among the processors. When splitting the domain by rows, for example, the number of active cells in each row are summed and summed again into a cumulative histogram. If the rows are to be divided into p groups, the cumulative histogram is searched for the splits closest to $1/p, 2/p, \ldots, p-1/p$. A similar search of the columns' active cell cumulative histogram is done to split the columns.

The decomposition described is used *only* for the cell-based calculations. The channel network is *not* divided among processes. Each process is assigned complete representation of the domain's entire channel network.



(a) (b)

Figure 2: DHSVM domain decomposition methods applied to a sample basin for 12 processors: splitting the domain into (a) groups of rows (`STRIPEY`) or (b) columns (`STRIPEX`). The default method is chosen depending on which global dimension is larger.

2.2.3. Input/Output Strategy

A distributed hydrology model like DHSVM requires considerable input data and can produce simulation results of considerable size. The choice of how the data are input and output can significantly affect parallel performance.

The input/output (I/O) strategy used here was relatively simple and largely emphasized maintaining existing behavior, such that the serial code structure was mostly maintained. When I/O bottlenecks are identified in future applications, a more complex strategy may be deployed.

All processes read the configuration file, so that, at startup, all processes have a complete description of the simulation without further communication. Other text files, like the stream network description, are also read by all processes. These files are typically small in size, and the time to read them is usually inconsequential.

DHSVM requires several input data sets that vary cell by cell. These data sets are input in the form of a two-dimensional (2D) raster map. In the parallel DHSVM, 2D map data are input through the root process (serially) then distributed via a global array. At the time of this writing, parallel I/O was not used, but may be supported in the future. For this work, DHSVM required considerable reworking of 2D data I/O to be able to work efficiently over a wide range of computational resources.

Two 2D map resolutions are necessary. The first is at the resolution of the DHSVM cell size and contain a single value for each cell. Data sets input at this resolution include the DEM, soil type, and vegetation type. Maps of this resolution are *partitioned* and distributed to processes according to the domain decomposition. The second map resolution is much coarser and not necessarily aligned with DHSVM cell boundaries. This was used for input of meteorological data fields. For data sets at this coarser resolution, the entire 2D map is *mirrored* on all processes, i.e., all processes receive an identical copy of the map. Mirroring the entire map in this way avoids a more complicated decomposition that would require overlapping sub-domains.

Figure 3 shows a schematic of I/O for 2D maps that are partitioned as they are distributed. A global array is created with a size to store values for the entire domain. A single process opens and reads, serially, a 2D map for the entire domain into local

memory. This process then puts[1] those data in the global array. All processes, including the process reading the map data, then get that portion of the global array it has been assigned. For this work, DHSVM required considerable reworking of 2D data I/O to be able to work efficiently over a range of computational resources. At the time of writing, parallel I/O was not used, but may be supported in the future.

Figure 4 shows a schematic of the input of maps that are mirrored across all processes. The input process is nearly the same as in Figure 3 except that each compute process gets the entire map and loads it into local memory.

In this work, meteorological data were supplied as a series of 2D maps of each required fields (as discussed in more detail below). At the beginning of each time step, the maps for that time step are read as described above.

All output, both 2D map data and text files, is through the root process. Writing 2D map data is the reverse of reading (Figure 3). Each process puts its local values in the global array, the root process gets the entire set of values and writes them to a file. Output other than 2D maps, (e.g., mass balance summary) requires a more traditional MPI-like all-reduce operation (using the GA API though).

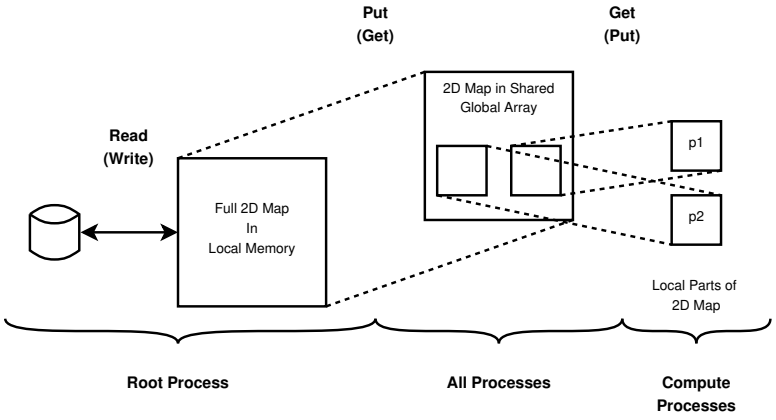


Figure 3: Schematic depicting the reading (writing) of *partitioned* 2D map data and their distribution to (from) compute processes. See text for further details.

2.2.4. Hydrologic Processes Adaptation

Figure 5 shows a simplified depiction of the parallel algorithm for a single simulation time step. Each simulation time step starts with time-step initialization (TSI). Several things are initialized at the beginning of a time step. Each process prepares the cells it owns for the next time step. The most important part of TSI is the assignment of meteorological data to individual cells. DHSVM has several available approaches to make this assignment, but each of eight meteorological data fields were read as *mir-*

[1]Here "put" and "get" are operations defined by the GA API. Another operation, "accumulate", is mentioned below.

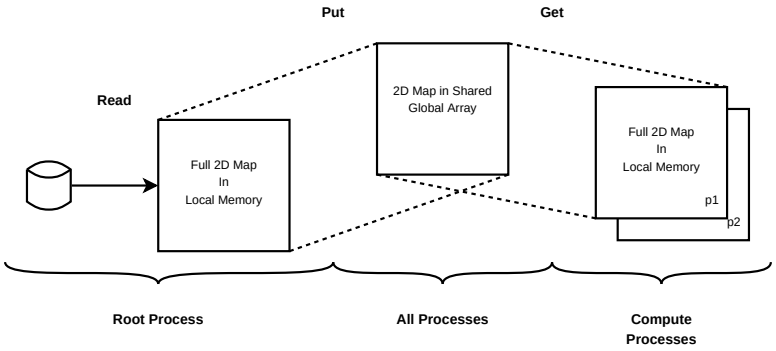


Figure 4: Schematic depicting the reading of *mirrored* 2D map data and their distribution to compute processes. See text for further details.

rored 2D maps (Section 2.2.3). This is a significant amount of data that needs to be read every time step.

Once cells are initialized, energy/water balance (EWB) calculations proceed. Each process updates the hydrologic and thermal state of the snowpack, vegetation canopy, and soil layers (Section 2.1.1) within the active cells assigned to it. The computations for a single cell do not require any communication with its neighbors, so this part of the simulation is most amenable to parallelization.

Unlike the EWB, subsurface and surface routing (SSR and SR) calculations (Section 2.1.2) require interaction with neighboring cells, and that interaction needed to extend between processors when neighboring cells were not owned by the same processor. Surface and subsurface routing have very similar algorithms, so the parallelization of those processes is handled in a similar manner. Figure 6 depicts the inter-process communication used for SSR and SR. The key issue with these processes is that a cell assigned to one processor may drain to a cell on another processor. This is handled by extending the calculated local domain by one cell. A temporary array is created on a local processor to hold the results of SR or SSR routing (Equations (1) and (2), respectively), as shown in Figure 6a. The array is sized to be one cell larger, in all (valid) directions, than the domain assigned to the processor (shaded gray in Figure 6a). That extra cell captures SSR or SR flux to the off-processor cell(s). As routing calculations proceed, surface water is routed to a cell outside the processor's domain, and the result is stored on the edge of the array.

After all processes complete local SR and SSR calculations, a global array for the entire domain is initialized to zero (Figure 6b). Each process *accumulates* the local array of routing results into the global array. In this way, water routed outside of the processors local domain is correctly captured and delivered to the neighboring domain. A get operation (Figure 6c) returns complete SR or SSR routing results from the global array to each processor's local memory.

Part of the SR and SSR algorithms is to compute the lateral inflow (L in Equations (3) and (4)) into each channel segment. However, each processor only computes lateral inflow contribution from the cells assigned it and it's necessary to add contributions from multiple processes. Consequently, lateral inflow for each channel segment

is totaled from the contributions computed by all processes. An all-reduce summation, typically an expensive operation, is used to sum lateral inflow over all processors. After the all-reduce, all processes have an identical array of lateral inflow to all segments.

The simulation time step ends with channel routing (CR, Section 2.1.3). Equations (3) and (4) require that a segment can only be routed after all upstream segments have been routed. Consequently, it was decided to keep CR a serial algorithm and that all processes would carry out identical computations. All processes then perform CR on the same network with the same inflow producing identical results. Only the root process outputs CR results.

An alternate approach would be to have the root process alone do the CR calculations, then distribute the CR results back to the other processors with a broadcast. The chosen approach avoids this second, possibly expensive, communication.

Other calculations are performed during a time step that are not depicted in Figure 5. The most important is a mass balance check. This requires each process to accumulate the mass balance components of its cells, then an all-reduce operation is used to sum the components over the domain.

2.3. Comparison to Serial DHSVM

Throughout the development of parallel DHSVM, it was periodically checked against the serial code, which was considered correct. several smaller basins were used to ensure that the serial and parallel codes produced identical results.

One such case was Rainy Creek. Rainy Creek is a small tributary (44 km^2) to the Wenatchee River in western Washington, USA. The Rainy Creek data set used in this comparison was that distributed as the DHSVM 3.0 tutorial[2]. Limited changes were made to configuration file(s) to conform to the current DHSVM code, but spatial and meteorologic data remained unchanged from the downloaded tutorial.

The basin was simulated for 18 months on an 8-core workstation running Linux. Simulation output from serial DHSVM code (`master` branch, commit 59e3230) was compared to that from the parallel code (`parallel` branch, commit e24b11c).

2.4. Case Studies

Two basins of different size were chosen to measure parallel DHSVM performance. Minimal simulation results are presented here, because the focus of this work is parallel performance. We emphasize, however, that these are real, detailed applications that use real data and are undergoing calibration and validation at the time of writing. The calibration, validation, and use of these cases will be documented in detail in future publications.

The Columbia River basin is located in the northwest continental United States and southern British Columbia, Canada (Figure 7) and it drains an area of 668,000 km^2 (Table 1). DHSVM was configured to simulate the Columbia River basin at a resolution of 90 m resulting in about 83 million active cells and 20,800 stream segments.

The Columbia River basin DEM (Figure 7) was derived from U.S. Geological Survey DEM (USGS, 2017) and Canada DEM (CDEM, NRC, 2015). Soil types were

[2] `https://dhsvm.pnnl.gov/tutorials.stm`

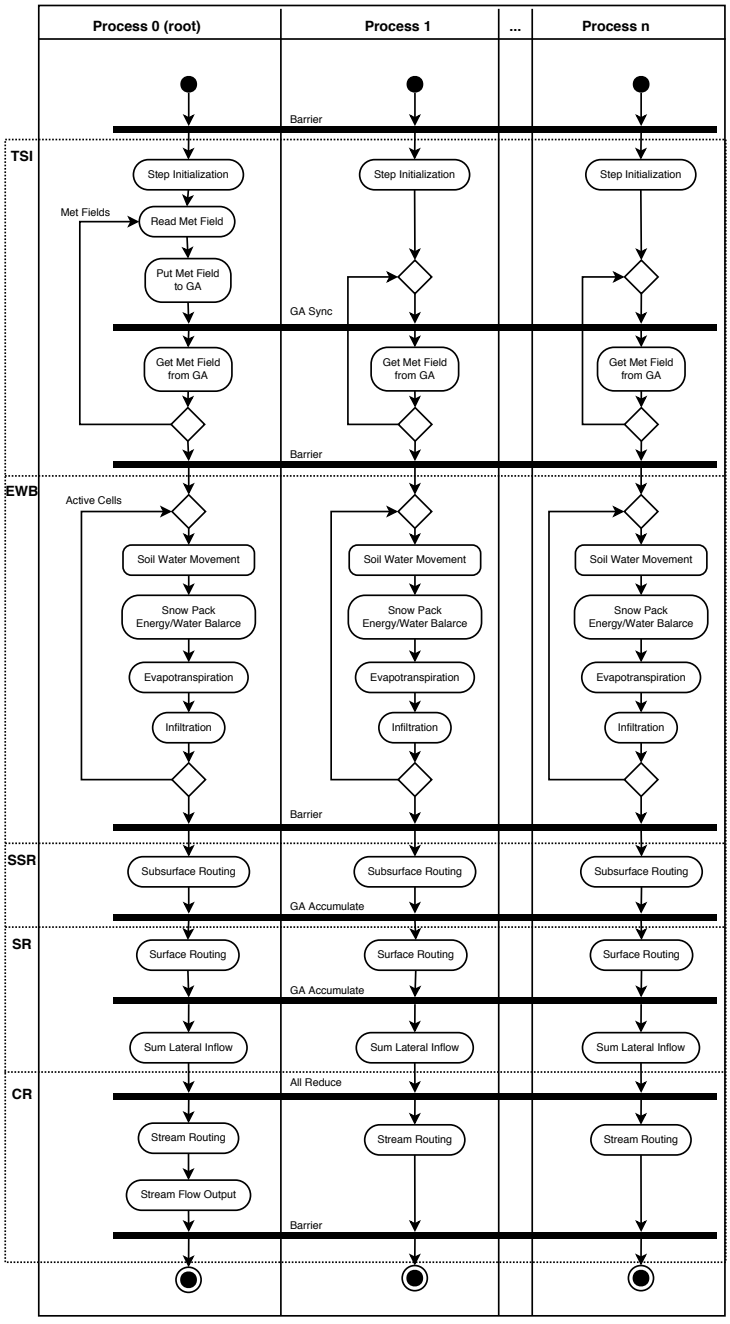


Figure 5: Simplified activity diagram for a single time step in parallel DHSVM. The dashed boxes indicate specific tasks discussed in the text: time-step initialization (TSI), energy/water balance (EWB), subsurface routing (SSR), surface routing (SR), and channel routing (CR).

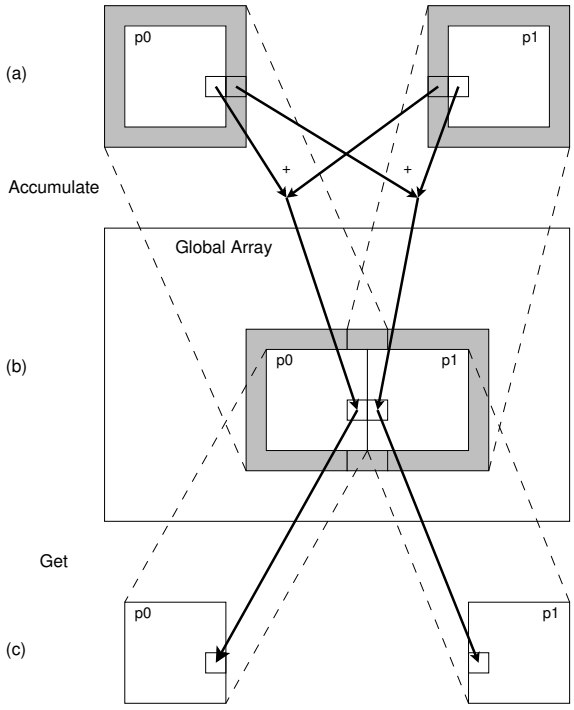


Figure 6: Schematic depicting inter-process communication needed for surface and subsurface routing. See text for discussion.

taken from Soil Survey Geographic Database (SSURGO, NRCS, 2019a), Digital General Soil Map of the United States (STATSGO2, NRCS, 2019b), and National Soil Database (CanSIS, 2014). Vegetation land cover was derived from the National Land Cover Data set (USGS, 2014) and Earth Observation for Sustainable Development of Forests (EOSD, Wood et al., 2002). The Columbia River basin stream network (Figure 8) was generated using the Python-based DHSVM pre-processing module, which calculates and extracts accumulated flow lines based on flow direction as derived from the DEM.

The Columbia River basin was simulated in two ways. In addition to complete simulation mode, DHSVM's "snow-only" mode was also used. In snow-only mode, DHSVM does not perform runoff-related computations, but instead concentrates on snowpack accumulation and melt. This is useful in snow-dominated applications because it allows calibration and validation of the snowpack simulation at a significantly lower computational cost.

The Clearwater River is an upland tributary in the Columbia River basin located in northern Idaho, USA (Figure 7). The basin area is 25,000 km^2, about 4% of the Columbia River basin (Table 1), and it produces about 7.5% of the Columbia River basin's average annual discharge. The Clearwater River basin DHSVM application was extracted as a subset of the Columbia River basin, so it has the same computational resolution, 90 m, and uses the same source data. The Clearwater application consisted of about 3 million active cells and 2,600 stream segments.

Table 1: Case study basin and DHSVM application statistics.

Basin Name	**Drainage Area** (km^2)	**Annual Disch.** (m^3/s)	**DHSVM Res.** (m)	**Region Rows** x **Cols** (cells)	**Active Cells**	**Stream Segments**
Columbia	668,000	5,850	90	14,599 x 12,654	83M	20,800
Clearwater	25,000	433	90	2,042 x 2,371	3M	2,600

2.4.1. Meteorological Forcing

In the case studies described above, meteorological forcings were generated using the Weather Research and Forecasting (WRF) model (Skamarock et al., 2008). The Advanced Research WRF Version 3.8 was used in this study, and the model was configured using physics options identical to those used by Gao et al. (2017). The historical climate simulation covered the period of 1981–2015, and WRF was driven by the large-scale circulations of the North American Regional Reanalysis (Mesinger et al., 2006) at 32 km grid resolution. The details of this simulation are provided by Chen et al. (2018). Meteorological forcings for DHSVM (precipitation, humidity, air temperature, wind speed, shortwave and long wave radiation) were archived hourly and transformed into the geographic projection used by DHSVM.

Within DHSVM, all data fields were assumed to be spatially constant over each 6 km cell with the exception of temperature and precipitation. Despite relatively high-resolution WRF data, the 6 km grids were still insufficient to capture some of the spatial variance of precipitation inside a WRF grid given the DHSVM resolution

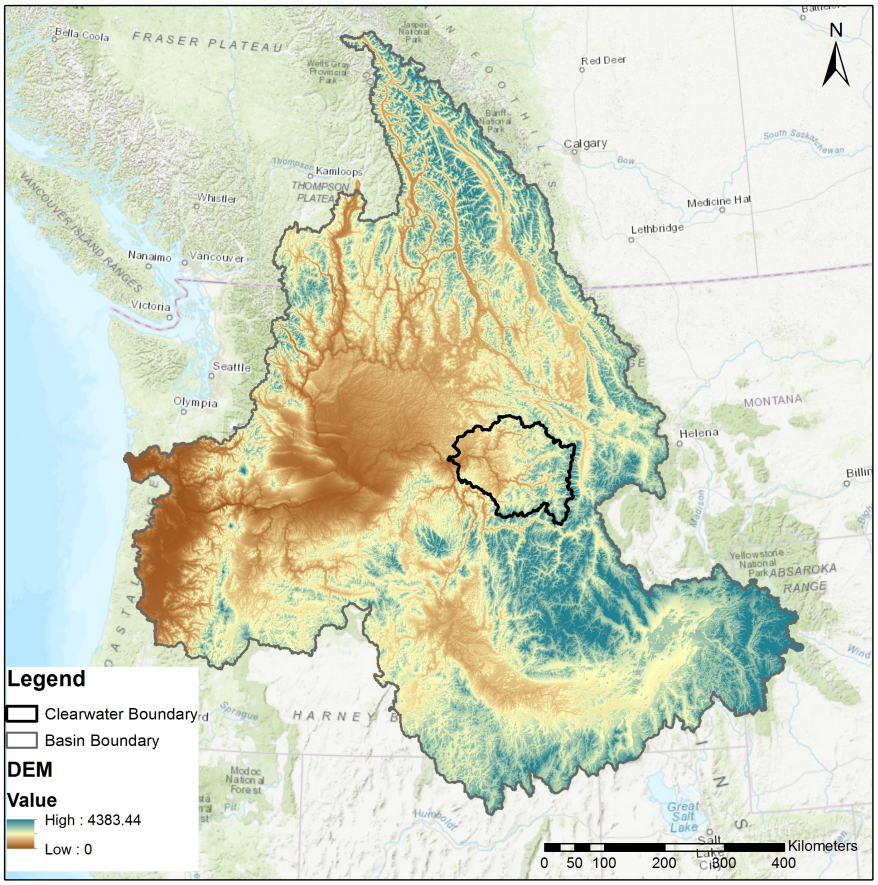


Figure 7: Map of elevation data (m) used for the Columbia River basin, consisting of 83,000,000 90x90 m cells. The Clearwater basin used the subset, 3,000,000 cells, outlined in black. Source: U.S. Geological Survey (USGS, 2017) for the United States and National Resources Canada (NRC, 2015) for Canada

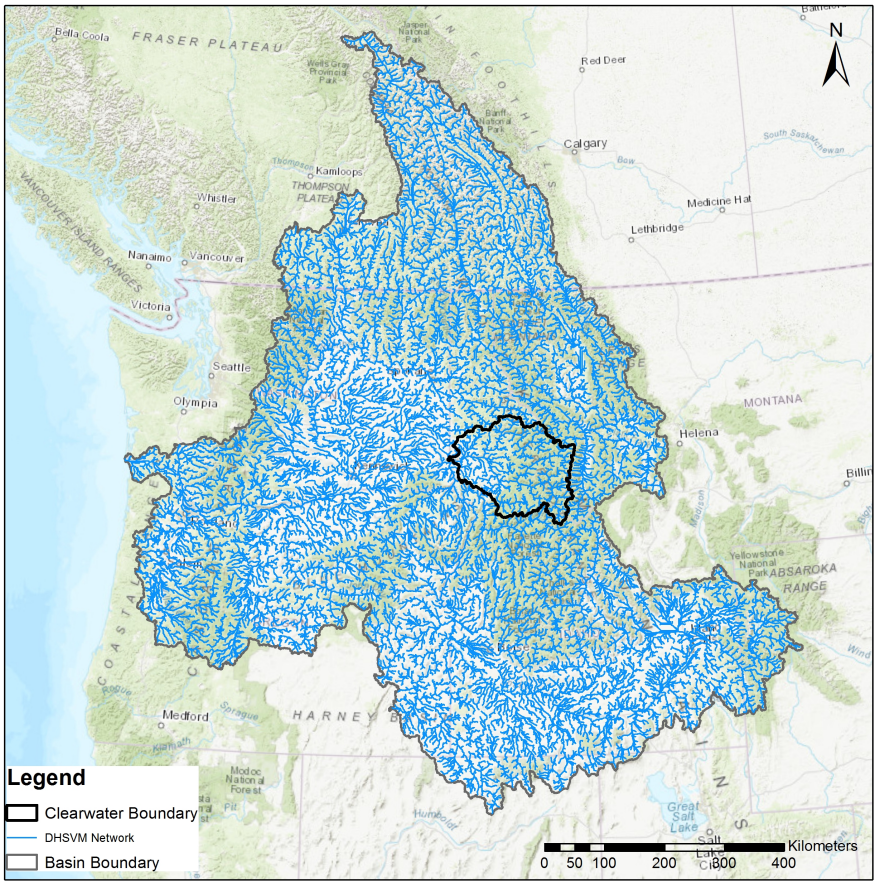


Figure 8: Columbia River basin stream network used in DHSVM, consisting of about 20,800 stream segments. The Clearwater basin used the 2,600 segments in the area outlined in black. The network was generated using the DHSVM stream network preprocessing module.

of 90 m. Therefore, WRF precipitation was down-scaled by introducing a monthly adjustment ratio at each DHSVM grid cell based on the Parameter-elevation Regressions on Independent Slopes Model (PRISM; PCG, 2004) precipitation data. Temperature was lapsed over each 6 km cell based on the elevation at each DHSVM grid cell and the elevation of the WRF cell. WRF to DHSVM down-scaling details will be documented in future publications.

WRF model results were prepared for a region large enough to encompass the entire Columbia River basin (220×191 6×6 km cells), using one file for each of the 6 variables containing maps for every hour of the simulation (1 year = 8760 hours). From those results, a subset was also extracted that covered the Clearwater River basin (32×37 6×6 km cells).

2.5. Parallel Performance

The two basins were simulated repeatedly for the 1982 water year with varying numbers of processors and the execution time was recorded. The simulations were carried out on the Pacific Northwest National Laboratory Institutional Computing (PIC) Constance cluster. This cluster consists of 528 compute nodes each having 24 cores (Intel Haswell E5-2670) and 64 GB of RAM. The nodes use FDR Infiniband connectivity. DHSVM and GA were built with Intel version 15 compilers and Intel MPI implementation. GA was built to use Infiniband connectivity directly (rather than MPI). At the time of this writing, this was considered to be the fastest transport layer for GA on systems using Infiniband interconnect.[3] The main disk storage system PIC cluster is a Lustre file system consisting of 42 Lustre object storage server (OSS) nodes with one object storage target per OSS and a capacity of 3.5 petabytes.

Simulation times were used to compute parallel speedup, defined as

$$s = \frac{T_s}{T_N} \tag{5}$$

where T_s is the execution time for a given problem on a single processor and T_N is the execution time for the same problem on N processors. Also computed was parallel efficiency,

$$e = \frac{T_s}{NT_N} \tag{6}$$

To provide key diagnostics, DHSVM was instrumented to report execution times of important tasks (some of these tasks are described in detail in Section 2.2.4):

Startup (SU): The SU task is all activity prior to simulating the first time step. The computational cost consists of memory allocation and initialization of the domain state. During this task, considerable input, including the DEM and related data and initial model state, is read and distributed among processors as partitioned 2D maps (Section 2.2.3). Domain decomposition (Section 2.2.2) is carried out during this task.

[3] Bruce Palmer, lead GA developer, personal communication.

Time-Step Initialization (TSI): The TSI task performs all necessary preparation for a new time step, the most costly of which, in this case, was the reading and distribution of meteorological data as mirrored 2D maps (Section 2.2.3).

Energy/Water Balance (EWB): EWB is the main computational task and was easily parallelized. Little or no interprocess communication or I/O happens during this task.

Subsurface Routing (SSR): The SSR task should be a mostly computational, where cell subsurface fluxes (Equation (2)) are computed for the cells a processor owns. The communication cost is the accumulation of a flux array over all processors.

Surface Routing (SR): As with SSR, SR computational cost is flux computation (Equation (1)) and communication cost is flux accumulation.

Channel Routing (CR): In the CR task, all processes perform identical routing calculations on the entire network, which means that the routing has a fixed computational cost regardless of the number of processors used. The main communication cost of this task is an all-reduce operation required to sum lateral inflow to the channel network across all processors.

Output (OUT): This task includes all significant output, except stream flow. Minimal output was specified for these simulations. Output of 2D maps was not specified. Consequently, the major cost of this task was the computation and output of an overall mass balance.

For the purposes of discussion, these tasks are grouped into primarily computational tasks (EWB, SSR, SR, CR) that are dominated by numerical computations. Communication tasks (SU, TSI, OUT) are those dominated by I/O and/or interprocess communication.

3. Results

3.1. Comparison to Serial DHSVM

Figure 9 visually compares Rainy Creek snow water equivalent depth and basin discharge simulated the parallel DHSVM, using 4 processors, to the current serial version of DHSVM. The parallel DHSVM simulation output was identical to that produced by the serial code to 5 significant figures. DHSVM generates a summary mass balance at the end of a simulation. Mass balance components are summed each time step over all active cells. At the end of the simulation, a mass balance error is computed. Table 3.1 compares this mass balance summary for serial and parallel simulations of Rainy Creek. Here too, the mass balance components were consistent to 5 significant figures. DHSVM computations are all single precision, so the differences between the simulations in Table 3.1 are easily explained by rounding and truncation errors.

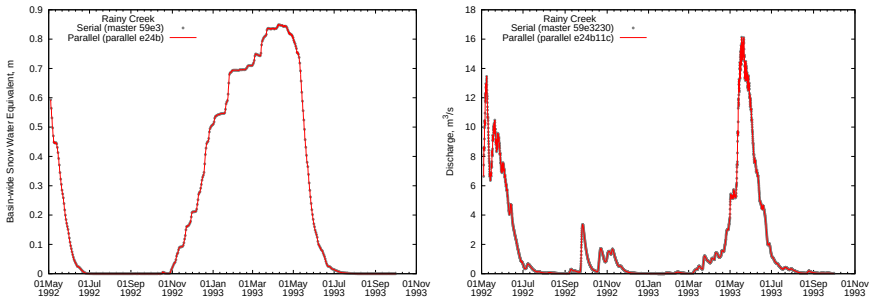


Figure 9: Comparison of serial and parallel DHSVM (4-cores) simulation results for Rainy Creek: basin-wide snow water equivalent (left) and basin outflow (right).

Table 2: DHSVM simulation final mass balances for 18-month simulations of Rainy Creek using serial DHSVM and parallel DHSVM with 4 and 8 processors. All numbers are mm depth over the entire basin.

		Parallel	
Mass Balance Component	**Serial**	**4 cores**	**8 cores**
Total Inflow	1917.402	1917.399	1917.398
Precipitation	1961.360	1961.357	1961.356
Snow Vapor Flux	-43.958	-43.958	-43.958
Total Outflow	2618.373	2618.376	2618.376
Evapotranspiration	628.706	628.708	628.708
Channel Interception	1989.667	1989.667	1989.668
Storage Change	-700.697	-701.010	-700.967
Initial Storage	997.251	997.451	997.415
Final Storage	296.553	296.441	296.448
Final Snow Water Equivalent	0.000	0.000	0.000
Final Soil Moisture	296.553	296.441	296.448
Final Surface	0.000	0.000	0.000
Final Road Surface	0.000	0.000	0.000
Mass Error	0.274	-0.033	0.012

3.2. Hydrologic Simulation

At the time of this writing, calibration and validation were under way for the Columbia River basin, of which the Clearwater River basin is a significant part. That will be documented in subsequent publications. The original serial DHSVM was not able to simulate either of these cases, mainly because of the problem size, but also because of some alterations that were not back-ported to the serial code. For this work, which focuses on computational performance, sufficient hydrological results are presented here to demonstrate that parallel DHSVM simulations show acceptable agreement with observations.

The Columbia River and nearly all of its larger tributaries have some kind of regulation and/or irrigation withdrawals. Because DHSVM, at this time, cannot represent this regulation, DHSVM is being calibrated and validated against estimates of no regulation, no irrigation (NRNI) streamflow for various locations in the Columbia River basin (BPA, 2011).

Figure 10 compares preliminary simulated Columbia River daily average discharge with NRNI at The Dalles for water years 1981 through 1985. The Dalles is significant gage location on the Columbia River approximately 300 km from the mouth, often used as representative of the basin. Figure 11 shows similar comparisons for discharges from some significant tributaries. The Clearwater and Salmon Rivers are tributaries to the Snake River. The Willamette River a large tributary that enters the Columbia River below The Dalles. The Pend Oreille River enters the Columbia near the U.S.-Canada border. All time series shown were extracted from the same simulation of the entire Columbia Basin. These time series show that the Columbia basin simulation has produced reasonable results at several locations, and much better matches can be expected with calibration. Figure 12 shows simulated snow water equivalent (SWE) depths over the entire Columbia River basin on April 1, 1982.

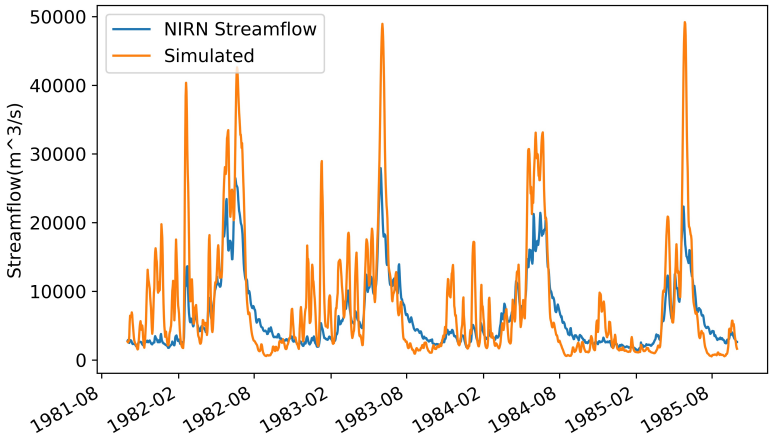


Figure 10: Preliminary comparison of no regulation, no irrigation adjusted streamflow (BPA, 2011, labeled "NRNI Streamflow") and simulated streamflow (labeled "WRF") in the Columbia River at The Dalles.

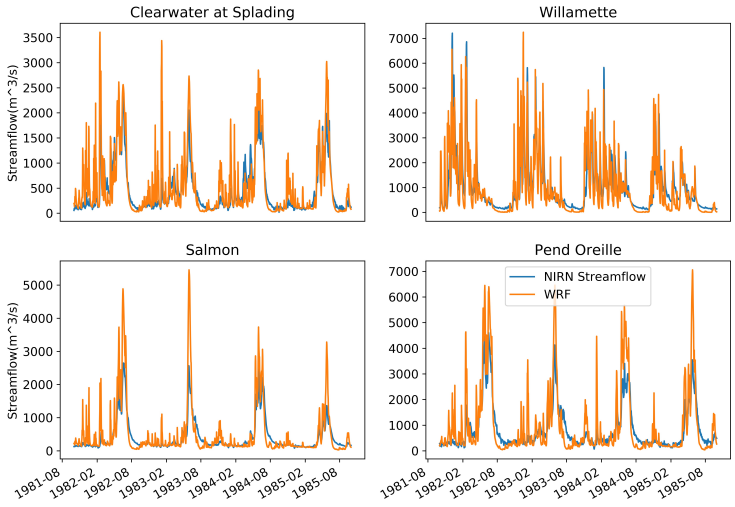


Figure 11: Preliminary comparison of no regulation, no irrigation adjusted streamflow (BPA, 2011, labeled "NRNI Streamflow") and simulated streamflow (labeled "WRF") from several significant tributaries to the Columbia River.

3.3. Parallel Performance

With the Clearwater simulation, a maximum speedup of about 32 was attained using 128 processors (Figure 16), about 23,000 active cells per processor. One year's simulation time was reduced from almost 4 hours using a single processor to 8 minutes using 128 processors (Table 3). With one processor, run time was dominated by computational tasks: EWB (70%), SR (7%), and SSR (13%) (Table 3, Figure 13). Computational tasks took a similar fraction of run time until about 32 processors were used. At this point, I/O tasks, specifically SU and TSI, started to take a larger fraction. This is the point at which parallel efficiency dropped quickly (Figure 17). At maximum speedup (128 processors), run time was split with about 40% for computational and 60% for I/O tasks.

The Columbia River basin simulations required a minimum of four nodes to run, due to the large memory requirements of the application. The smallest number of cores we could use was four. Run time for one processor was assumed to be four times that of four processors, so that between one and four processors speedup assumed to be ideal and parallel efficiency was assumed to be 1.0.

With the Columbia snow-only simulation, the maximum speedup was about 93 using 480 processors, at about 173,000 active cells per processor (Figure 16). The 1-year simulation time was reduced from about 10 days using one processor (estimated) to 2.5 hours using 480 processors. With four processors, the run time was dominated by EWB (80%), and that task dominates until 120 processors are used (56%) when the main I/O tasks (SU and TSI) begin to dominate. At maximum speedup, EWB takes about 29% and SU and TSI take about 63% of the simulation time (Table 3, Figure 14).

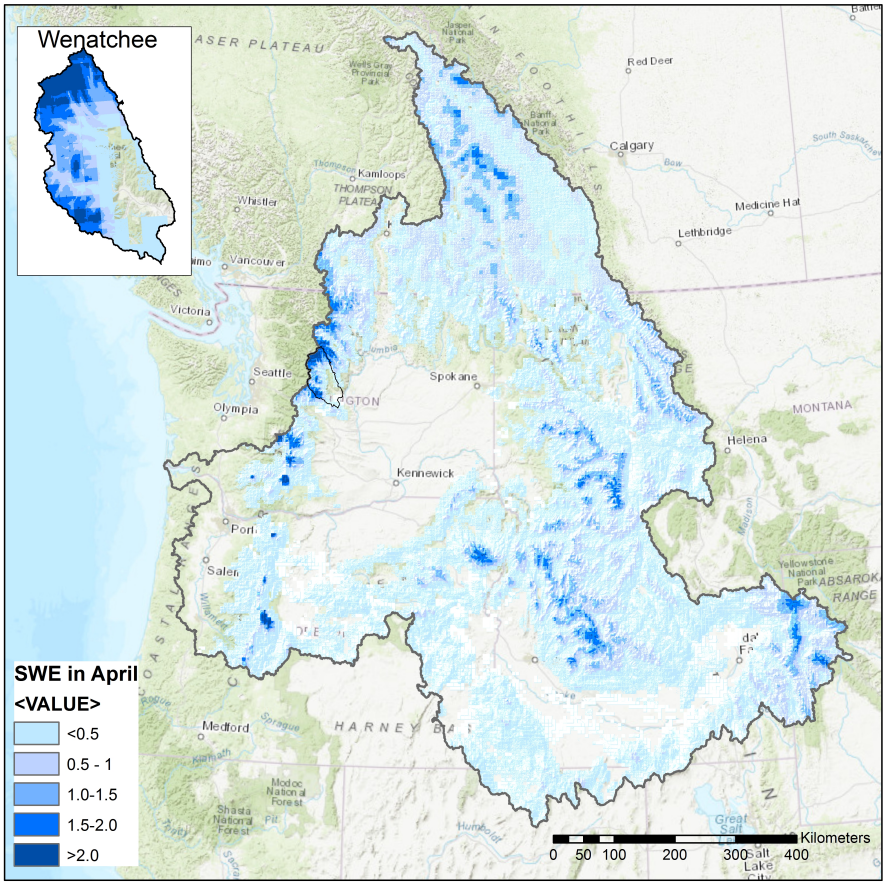


Figure 12: Simulated snow water-equivalent (SWE) at 90 m resolution for the Columbia River basin.

Maximum speedup for the full Columbia simulation, which included the water routing tasks, was about 105 also using 480 processors (Figure 16). The 1-year simulation time was reduced from about 19 days with one processor (estimated) to about 4 hours (Table 3) with 480 processors. With four processors, run time was dominated by computational tasks (90%), with EWB dominating that (67%). At maximum speedup (480 processors), the run time was split with about 60% for computational and 40% for I/O tasks. This is the reverse of the split for the Clearwater at maximum speedup. Note that CR took a much larger part of the simulation time for the Columbia (21%) than for the Clearwater (5%).

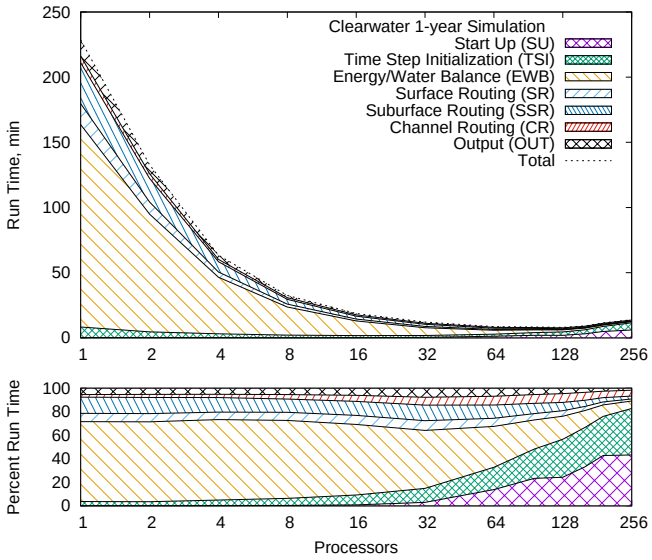


Figure 13: DHSVM timing results, including specific tasks, for a 1-year simulation of the Clearwater River basin. See the text for descriptions of timed tasks.

4. Discussion

In this work, we modified DHSVM to run in parallel using GA for interprocess communication targeting large, distributed memory systems. Simulation run times for our test cases were reduced enough to make long-term (decades), high-spatial resolution simulations of significantly sized basins manageable. As expected, the run times with low numbers of processes were dominated by the computational tasks, namely EWB, SR, and SSR. IO-intensive tasks, namely SU and TSI, become dominant at higher core counts, indicating more interprocess communication.

It was not straightforward to compare our results with those in the literature. Different models use different methods that have different computational and communication costs. Our speedup was on par with that measured by Vivoni et al. (2011) and not nearly as good as that of Kumar and Duffy (2016), the only other examples we could find using a parallel, distributed hydrology model running on 100s of processors. A previous

Table 3: DHSVM run times for a 1-year simulation using varying numbers of processors. Timed tasks are described in the text: startup (SU), time-step initialization (TSI), energy/water balance (EWB), subsurface routing (SSR), surface routing (SR), channel routing (CR), and output (OUT). *s* and *e* are parallel scale up (Equation (5)) and efficiency (Equation (6)), respectively.

	Run Time	**Percent Run Time per Task**								
NP	(min)	**SU**	**TSI**	**EWB**	**SR**	**SSR**	**CR**	**OUT**	*s*	*e*
Clearwater										
1	228.5	0.1	3.6	67.9	7.1	13.7	2.2	5.4	1.0	1.00
2	132.8	0.1	3.4	68.0	7.0	13.7	2.7	5.2	1.7	0.86
4	63.4	0.2	4.7	68.4	6.3	12.4	2.8	5.1	3.6	0.90
8	32.5	0.3	6.1	66.2	6.9	11.2	3.8	5.4	7.0	0.88
16	18.7	0.8	8.4	60.0	7.7	12.0	5.1	6.0	12.2	0.76
32	12.1	3.0	11.9	49.2	8.3	13.3	6.7	7.7	18.9	0.59
64	8.6	13.8	19.0	34.7	6.9	11.3	7.4	6.8	26.5	0.41
96	8.3	23.4	24.9	24.7	5.4	8.7	7.4	5.4	27.4	0.29
128	8.0	24.3	32.4	19.6	4.7	7.1	7.5	4.6	28.5	0.22
160	9.2	33.2	33.6	14.4	3.5	5.2	6.7	3.5	24.9	0.16
192	11.5	42.7	33.3	9.7	2.6	3.6	5.6	2.5	19.9	0.10
256	13.9	43.2	39.6	6.3	1.7	2.4	5.0	1.7	16.4	0.06
Columbia, Snow-Only Mode										
4	3498.9	0.1	10.7	81.3	0.0	0.0	0.0	8.0	4.0	1.00
8	2038.6	0.1	12.8	79.2	0.0	0.0	0.0	7.8	6.9	0.86
16	1173.1	0.3	16.7	75.3	0.0	0.0	0.0	7.8	11.9	0.75
32	707.3	0.3	20.3	71.5	0.0	0.0	0.0	7.9	19.8	0.62
64	436.1	2.5	24.2	64.3	0.0	0.0	0.0	9.0	32.1	0.50
120	281.9	4.4	28.8	56.4	0.0	0.0	0.0	10.5	49.6	0.41
240	169.2	3.1	38.0	48.7	0.0	0.0	0.0	10.2	82.7	0.34
480	150.3	12.8	50.4	28.5	0.0	0.0	0.0	8.3	93.1	0.19
720	165.5	14.5	60.4	17.7	0.0	0.0	0.0	7.4	84.6	0.12
960	187.5	13.7	66.2	12.4	0.0	0.0	0.0	7.7	74.6	0.08
Columbia										
4	6713.3	0.0	5.6	66.5	7.0	13.0	3.5	4.5	4.0	1.00
8	3710.4	0.1	7.1	64.0	6.7	12.4	4.7	5.1	7.2	0.90
16	2136.7	0.1	9.1	58.2	7.4	12.9	6.6	5.8	12.6	0.79
32	1236.9	0.2	11.7	54.9	7.5	13.3	6.8	5.6	21.7	0.68
64	800.4	0.3	13.3	46.9	8.4	14.6	9.6	6.9	33.5	0.52
120	530.2	0.8	15.4	39.6	8.6	13.7	14.3	7.7	50.6	0.42
240	335.4	2.5	19.2	32.0	7.5	13.7	17.6	7.5	80.1	0.33
480	255.1	3.8	30.1	21.6	6.0	10.5	20.9	7.1	105.3	0.22
720	272.3	6.0	37.1	14.3	5.3	9.5	20.6	7.2	98.6	0.14
960	291.4	10.0	43.7	10.0	3.9	6.7	18.3	7.3	92.2	0.10

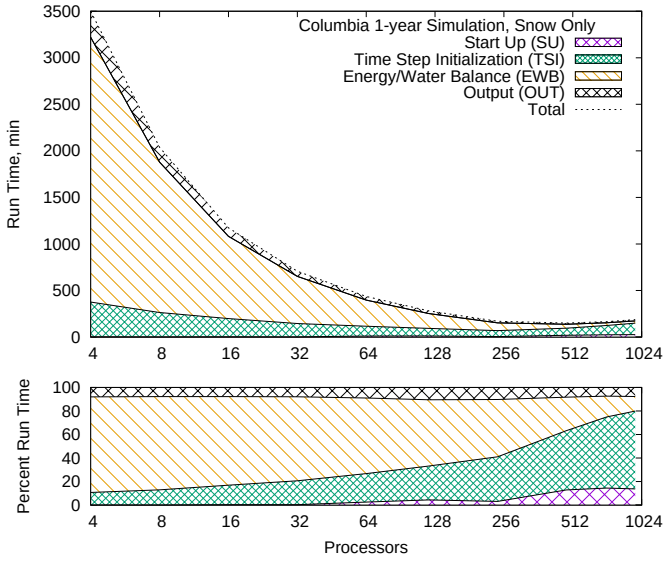


Figure 14: DHSVM simulation timing results, including specific tasks, for the Columbia River basin in snow-only mode. See the text for descriptions of timed tasks.

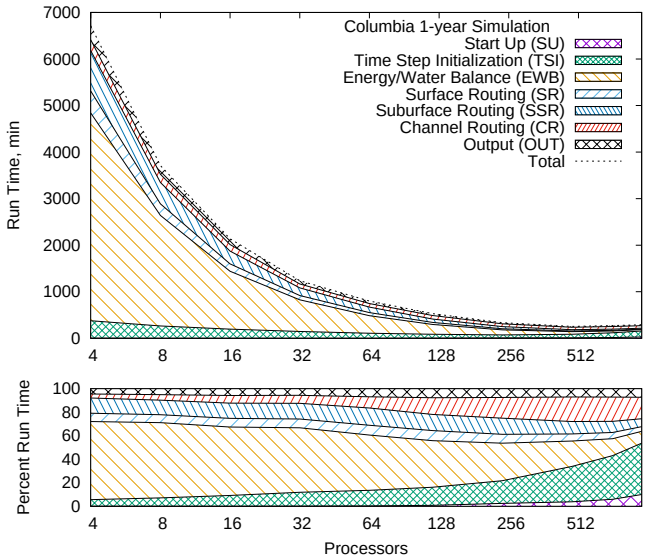


Figure 15: DHSVM simulation timing results, including specific tasks, for the Columbia River basin in normal simulation mode. See the text for descriptions of timed tasks.

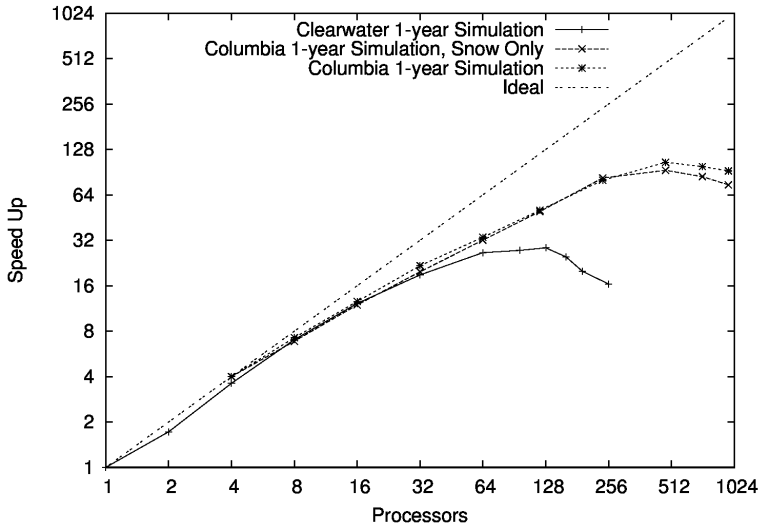


Figure 16: Measured DHSVM parallel speedup.

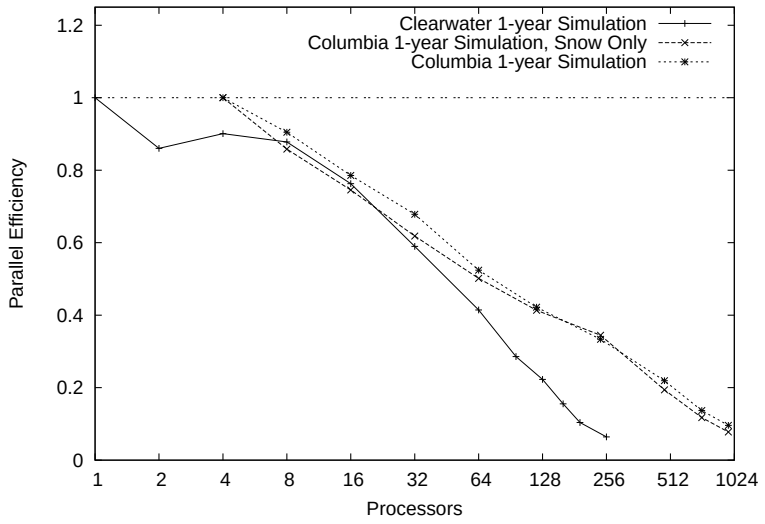


Figure 17: Computed DHSVM parallel efficiency.

effort with DHSVM (Adriance et al., 2019) attained speed ups of 440% using compiler optimization and OpenMP. The Kumar and Duffy application appears to be designed and implemented as a parallel application, as opposed to our retrofit of DHSVM. Our speedup was also comparable to some others (e.g., Liu et al., 2014, 2016) that used 10–20 processors, and had significantly smaller-sized problems.

Our approach is distinct from other approaches in two ways. The first is exclusive reliance on the GA one-sided communication API. Other parallel distributed hydrology models tend to use MPI on distributed memory platforms (e.g., Vivoni et al., 2011; Kumar and Duffy, 2016) or OpenMP on shared memory systems (e.g., Li et al., 2010, 2011; Hwang et al., 2014; Liu et al., 2014, 2016). DHSVM has an advantage here, in that it has a rectangular domain, which fits the data structure model of GA nicely. The use of GA expands the range of computational platforms that can be brought to bear on these hydrologic problems.

The second distinction is the straightforward domain decomposition technique. DHSVM domain decomposition simply divides rows or columns of the rectangular domain, unlike other approaches that divide the domain by drainage network (Apostolopoulos and Georgakakos, 1997; Grbsch and David, 2001; Li et al., 2011; Liu et al., 2016). The triangular cell networks used by Vivoni et al. (2011) and Kumar and Duffy (2016) required more complicated algorithms to decompose the domain. Our approach is simple and requires very little computational effort, but could perhaps be improved. While not explicitly investigated here, load balancing likely plays a significant role in DHSVM parallel performance and should be examined as part of further code improvements.

DHSVM parallel efficiency falls off as more processors are used (Figure 17). Decreasing efficiency is due to the parts of the simulation that are serial and require the same amount of time regardless of the number of processes involved. Parallel efficiency indicates the relative benefit of adding more processors to the calculation. As more processors are added, the fixed-time tasks take a larger proportion of the simulation time, which reduces the relative benefit of the additional processors. In this case, the most prominent culprits (as shown in Figure 15) are TSI (input) and CR. In order to get better efficiency, those parts of simulation need to either more efficient (i.e. take less time), or be parallelized in some way. Some improvements to these are being considered, as discussed below.

The parallel performance also indicates that running DHSVM at the point of maximum speedup may not be ideal. Run time needs to be balanced with the availability and cost of computational resources. For example, the Columbia simulation had a maximum speedup with 480 processors with a simulation time of about 4 hours. If the same simulation is run with 120 processors, it would take 8 hours. While the run time would be doubled, the computational cost would only be one quarter. Additionally, a set of 480 processors is most likely less available than 120, which may lead to longer job queue times. It may also be more efficient to simulate a case like the Columbia River basin in several large subbasins, particularly for calibration and validation. Once calibrated, the parameters could be used in a "production" simulation of the entire Columbia Basin.

In this initial parallel version of DHSVM, we emphasized maintenance of current capabilities and avoiding large structural changes to the code. Parallel performance

may have been limited by the emphasis on limiting code changes. This is particularly true with the input of 2D maps.

5. Future Work

The authors are generally pleased with the performance improvements attained with this work. DHSVM's parallel performance was good enough to tackle the task at hand, namely the entire Columbia River basin. However, a larger application at this ultra-fine 90 m resolution still may be challenging at this time.

The timings clearly indicate that the way meteorological forcing was read and applied was the largest single obstacle to higher parallel performance. Several ideas may be investigated to reduce this load, but they would probably require significant structural changes to the current DHSVM code. Reading the meteorological data in larger blocks, a day or month at a time, say, rather than one time step at a time, may reduce input time. Reading 2D map data in parallel, instead of through the root process (Section 2.2.3), may also be a solution.

Stream routing took a significant part of the total run time for the Columbia simulation. The choice to keep this a serial process, executed by all processes, may be acceptable for smaller basins, and was acceptable for the cases here, but may become a barrier with larger applications. Parallel methods to perform channel routing will be investigated in future work.

We have used a straightforward and relatively simple domain decomposition scheme here. A more extensive investigation of domain decomposition would likely yield further performance improvements. We have assumed that the simulation of each "active" cell has an equivalent computational cost. This is not strictly true. Cells with snow definitely have a higher computational cost than cells without snow. Such an investigation would require some detailed analysis of run times and how snow increases the computational cost of an active cell.

Acknowledgments

The study was primarily funded by the U.S. Department of Energy (DOE) Office of Energy Efficiency and Renewable Energy, Water Power Technologies Office. The material is also based upon work funded by the Strategic Environmental Research and Development Program under contract RC-2546 and the DOE Office of Science Biological and Environmental Research as part of the Regional and Global Climate Modeling and Multi-Sector Energy and Environmental Dynamics programs. The WRF simulations described here were performed using the facilities of the Pacific Northwest National Laboratory (PNNL) institutional computing center (PIC) and the National Energy Research Supercomputing Center, which is supported by the DOE Office of Science under contract DE-AC02-05CH11231. DHSVM simulations described here were also performed using PNNL PIC facilities. PNNL is operated by Battelle for the DOE under contract DE-AC06-76RLO-1830.

DHSVM is maintained jointly by the Hydrology Group at PNNL and the Civil Engineering Department at the University of Washington. More information can be

found at http://dhsvmdev.pnl.gov/. DHSVM source code can be obtained from its Github repository (https://github.com/pnnl/DHSVM-PNNL). The code used for this work is in the "parallel" branch.

References

Adriance, A., Pantoja, M., Lupo, C., 2019. Acceleration of Hydrology Simulations Using DHSVM for Multi-thousand Runs and Uncertainty Assessment, in: Meneses, E., Castro, H., Barrios Hernndez, C.J., Ramos-Pollan, R. (Eds.), High Performance Computing, Springer International Publishing. pp. 179–193.

Apostolopoulos, T.K., Georgakakos, K.P., 1997. Parallel computation for streamflow prediction with distributed hydrologic models. Journal of Hydrology 197, 1–24. doi:10.1016/S0022-1694(96)03281-7.

Bales, R.C., Molotch, N.P., Painter, T.H., Dettinger, M.D., Rice, R., Dozier, J., 2006. Mountain hydrology of the western United States. Water Resources Research 42. doi:10.1029/2005WR004387.

Beckers, J., Smerdon, B., Wilson, M., others, 2009. Review of hydrologic models for forest management and climate change applications in British Columbia and Alberta. Forrex series .

Bonneville Power Administration (BPA), 2011. 2010 Level Modified Streamflow 1928-2008. Technical Report DOE/BP-4352. Bonneville Power Administration. Portland, OR, USA.

Canadian Soil Information Service (CanSIS), 2017. National soil database (NSDB). Available online. URL: http://sis.agr.gc.ca/cansis/nsdb/index.html. Last accessed: 2019-01-31.

Cao, Q., Sun, N., Yearsley, J., Nijssen, B., Lettenmaier, D.P., 2016. Climate and land cover effects on the temperature of Puget Sound streams. Hydrological Processes 30, 2286–2304. doi:10.1002/hyp.10784.

Chen, X., Leung, L.R., Gao, Y., Liu, Y., Wigmosta, M., Richmond, M., 2018. Predictability of extreme precipitation in western U.S. watersheds based on atmospheric river occurrence, intensity, and duration. Geophysical Research Letters 45, 11,693–11,701. doi:10.1029/2018GL079831.

Cristea, N.C., Lundquist, J.D., Loheide, S.P., Lowry, C.S., Moore, C.E., 2014. Modelling how vegetation cover affects climate change impacts on streamflow timing and magnitude in the snowmelt-dominated upper Tuolumne Basin, Sierra Nevada. Hydrological Processes 28, 3896–3918. doi:10.1002/hyp.9909.

Cuo, L., Lettenmaier, D.P., Alberti, M., Richey, J.E., 2009. Effects of a century of land cover and climate change on the hydrology of the Puget Sound basin. Hydrological Processes 23, 907–933. doi:10.1002/hyp.7228.

Cuo, L., Lettenmaier, D.P., Mattheussen, B.V., Storck, P., Wiley, M., 2008. Hydrologic prediction for urban watersheds with the Distributed HydrologySoilVegetation Model. Hydrological Processes 22, 4205–4213. doi:10.1002/hyp.7023.

Dagum, L., Menon, R., 1998. OpenMP: An Industry-Standard API for Shared-Memory Programming. IEEE Comput. Sci. Eng. 5, 46–55. doi:10.1109/99.660313.

Dinan, J., Balaji, P., Hammond, J.R., Krishnamoorthy, S., Tipparaju, V., 2012. Supporting the global arrays PGAS model using MPI one-sided communication, in: 2012 IEEE 26th International Parallel and Distributed Processing Symposium, IEEE, Shanghai, China. pp. 739–750. doi:10.1109/IPDPS.2012.72.

Frans, C., Istanbulluoglu, E., Lettenmaier, D.P., Fountain, A.G., Riedel, J., 2018. Glacier Recession and the Response of Summer Streamflow in the Pacific Northwest United States, 19602099. Water Resources Research 54, 6202–6225. doi:10.1029/2017WR021764.

Frans, C., Istanbulluoglu, E., Lettenmaier, D.P., Naz, B.S., Clarke, G.K.C., Condom, T., Burns, P., Nolin, A.W., 2015. Predicting glacio-hydrologic change in the headwaters of the Zongo River, Cordillera Real, Bolivia. Water Resources Research 51, 9029–9052. doi:10.1002/2014WR016728.

Gao, Y., Leung, R.L., Zhao, C., Hagos, S., 2017. Sensitivity of U.S. summer precipitation to model resolution and convective parameterizations across gray zone resolutions. Journal of Geophysical Research: Atmospheres 122, 2714–2733. doi:10.1002/2016JD025896.

Grbsch, M., David, O., 2001. How to divide a catchment to conquer its parallel processing. An efficient algorithm for the partitioning of water catchments. Mathematical and Computational Modelling 33, 723–731. doi:10.1016/S0895-7177(00)00275-2.

Hwang, H.T., Park, Y.J., Sudicky, E.A., Forsyth, P.A., 2014. A parallel computational framework to solve flow and transport in integrated surfacesubsurface hydrologic systems. Environmental Modelling & Software 61, 39–58. doi:10.1016/j.envsoft.2014.06.024.

Kumar, M., Duffy, C.J., 2016. Exploring the role of domain partitioning on efficiency of parallel distributed hydrologic model simulations. Journal of Hydrogeology & Hydrologic Engineering 2015. doi:10.4172/2325-9647.1000119.

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

Leung, L.R., Wigmosta, M.S., 1999. Potential climate change impacts on mountain watersheds in the Pacific Northwest. Journal of the American Water Resources Association 35, 1463–1471. doi:10.1111/j.1752-1688.1999.tb04230.x.

Li, T., Wang, G., Chen, J., 2010. A modified binary tree codification of drainage networks to support complex hydrological models. Computers & Geosciences 36, 1427–1435. doi:10.1016/j.cageo.2010.04.009.

Li, T., Wang, G., Chen, J., Wang, H., 2011. Dynamic parallelization of hydrological model simulations. Environmental Modelling & Software 26, 1736–1746. doi:10.1016/j.envsoft.2011.07.015.

Liu, J., Zhu, A.X., Liu, Y., Zhu, T., Qin, C.Z., 2014. A layered approach to parallel computing for spatially distributed hydrological modeling. Environmental Modelling & Software 51, 221–227. doi:10.1016/j.envsoft.2013.10.005.

Liu, J., Zhu, A.X., Qin, C.Z., Wu, H., Jiang, J., 2016. A two-level parallelization method for distributed hydrological models. Environmental Modelling & Software 80, 175–184. doi:10.1016/j.envsoft.2016.02.032.

Livneh, B., Deems, J.S., Buma, B., Barsugli, J.J., Schneider, D., Molotch, N.P., Wolter, K., Wessman, C.A., 2015. Catchment response to bark beetle outbreak and dust-on-snow in the Colorado Rocky Mountains. Journal of Hydrology 523, 196–210. doi:10.1016/j.jhydrol.2015.01.039.

Manojkumar, K., Palmer, B., Vishnu, A., Krishnamoorthy, S., Daily, J., Chavarria, D., 2012. The Global Arrays User Manual. Technical Report PNNL-13130. Pacific Northwest National Laboratory. Richland, WA.

Mesinger, F., DiMego, G., Kalnay, E., Mitchell, K., Shafran, P.C., Ebisuzaki, W., Jović, D., Woollen, J., Rogers, E., Berbery, E.H., Ek, M.B., Fan, Y., Grumbine, R., Higgins, W., Li, H., Lin, Y., Manikin, G., Parrish, D., Shi, W., 2006. North American Regional Reanalysis. Bulletin of the American Meteorological Society 87, 343–360. doi:10.1175/BAMS-87-3-343.

MPI Forum, 2018. Message Passing Interface (MPI) Forum Home Page. http://www.mpi-forum.org/ (Sept. 2018).

Natural Resources Canada (NRC), 2015. Canadian digital elevation model (CDEM). Government of Canada. Available online. URL: https://open.canada.ca/data/en/dataset/7f245e4d-76c2-4caa-951a-45d1d2051333. Last accessed: 2019-01-31.

Natural Resources Conservation Service (NRCS), 2019a. Soil survey geographic (SSURGO) database. U.S. Department of Agriculture. Available online. URL: https://sdmdataaccess.sc.egov.usda.gov. Last accessed 2019-01-27.

Natural Resources Conservation Service (NRCS), 2019b. U.S. general soil map (STATSGO2). U.S. Department of Agriculture. Available online. URL: https://sdmdataaccess.sc.egov.usda.gov. Last accessed 2019-01-27.

Naz, B.S., Frans, C.D., Clarke, G.K.C., Burns, P., Lettenmaier, D.P., 2014. Modeling the effect of glacier recession on streamflow response using a coupled glacio-hydrological model. Hydrology and Earth System Sciences 18, 787–802. doi:10.5194/hess-18-787-2014.

Nieplocha, J., Palmer, B., Tipparaju, V., Krishnan, M., Trease, H., Aprà, E., 2006. Advances, applications and performance of the global arrays shared memory programming toolkit. The International Journal of High Performance Computing Applications 20, 203–231. doi:10.1177/1094342006064503.

PRISM Climate Group (PCG), 2004. Parameter-elevation regressions on independent slopes model. Oregon State University. Available online. URL: http://prism.oregonstate.edu. created 2004-02-04.

Simeone, B., 1986. An asymptotically exact polynomial algorithm for equipartition problems. Discrete Applied Mathematics 14, 283–293. doi:10.1016/0166-218X(86)90032-6.

Skamarock, W.C., Klemp, J.B., Dudhia, J., Gill, D.O., Barker, D.M., Duda, M.G., Huang, X.Y., Wang, W., Powers, J.G., 2008. A Description of the Advanced Research WRF Version 3. NCAR Technical Note NCAR/TN-475+STR. National Center for Atmospheric Research (NCAR). doi:10.5065/D68S4MVH.

Storck, P., Bowling, L., Wetherbee, P., Lettenmaier, D., 1998. Application of a GIS-based distributed hydrology model for prediction of forest harvest effects on peak stream flow in the Pacific Northwest. Hydrological Processes 12, 889–904. doi:10.1002/(SICI)1099-1085(199805)12:6<889::AID-HYP661>3.0.CO;2-P.

Storck, P., Lettenmaier, D.P., 1999. Predicting the effect of a forest canopy on ground snow accumulation and ablation in maritime climates, in: Proceedings of 67th Western Snow Conference, Colo. State Univ. Fort Collins. pp. 1–12.

Sun, N., Wigmosta, M., Zhou, T., Lundquist, J., DickersonLange, S., Cristea, N., 2018. Evaluating the functionality and streamflow impacts of explicitly modelling forest-snow interactions and canopy gaps in a distributed hydrologic model. Hydrological Processes 32, 2128–2140. doi:10.1002/hyp.13150.

Sun, N., Yearsley, J., Baptiste, M., Cao, Q., Lettenmaier, D.P., Nijssen, B., 2016. A spatially distributed model for assessment of the effects of changing land use and climate on urban stream quality. Hydrological Processes 30, 4779–4798.

Sun, N., Yearsley, J., Voisin, N., Lettenmaier, D.P., 2015. A spatially distributed model for the assessment of land use impacts on stream temperature in small urban watersheds. Hydrological Processes 29, 2331–2345.

Thyer, M., Beckers, J., Spittlehouse, D., Alila, Y., Winkler, R., 2004. Diagnosing a distributed hydrologic model for two high-elevation forested catchments based on detailed stand- and basin-scale data. Water Resources Research 40. doi:10.1029/2003WR002414.

U.S. Geological Survey (USGS), 2014. NLCD 2011 land cover (2011 edition, amended 2014) - national geospatial data asset (NGDA) land use land cover. URL: https://www.sciencebase.gov/catalog/item/4f70a43ce4b058caae3f8db3. Last accessed: 2019-01-31.

U.S. Geological Survey (USGS), 2017. 1 arc-second digital elevation models (DEMs) - USGS national map 3DEP downloadable data collection. Available online. URL: https://www.sciencebase.gov/catalog/item/4f70aa71e4b058caae3f8de1. Last accessed: 2019-01-31.

Vivoni, E.R., Mascaro, G., Mniszewski, S., Fasel, P., Springer, E.P., Ivanov, V.Y., Bras, R.L., 2011. Real-world hydrologic assessment of a fully-distributed hydrological model in a parallel computing environment. Journal of Hydrology 409, 483–496. doi:10.1016/j.jhydrol.2011.08.053.

Wigmosta, M.S., Lettenmaier, D.P., 1999. A comparison of simplified methods for routing topographically driven subsurface flow. Water Resources Research 35, 255–264. doi:10.1029/1998WR900017.

Wigmosta, M.S., Storck, P., Lettenmaier, D.P., 2002. The distributed hydrology soil vegetation model, in: Mathematical Models of Small Watershed Hydrology and Applications. Water Resource Publications, Littleton, CO, pp. 7–42.

Wigmosta, M.S., Vail, L.W., Lettenmaier, D.P., 1994. A distributed hydrology-vegetation model for complex terrain. Water Resources Research 30, 1665–1679. doi:10.1029/94WR00436.

Wilkinson, B., Allen, M., 1998. Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers. Prentice Hall.

Wood, J.E., Gillis, M.D., Goodenough, D.G., Hall, R.J., Leckie, D.G., Luther, J.E., Wulder, M.A., 2002. Earth observation for sustainable development of forests (EOSD): Project overview, in: IEEE International Geoscience and Remote Sensing Symposium, pp. 1299–1302 vol.3. doi:10.1109/IGARSS.2002.1026097.