Kumar and Duffy. J Hydrogeol Hydrol Eng 2015, 4:1
http://dx.doi.org/10.4172/2325-9647.1000119

Journal of Hydrogeology & Hydrologic Engineering

Research Article

A SCITECHNOL JOURNAL

Exploring the Role of Domain Partitioning on Efficiency of Parallel Distributed Hydrologic Model Simulations

Mukesh Kumar[1*] and Christopher J Duffy[2]

Abstract

Spatially distributed hydrologic models of watersheds and river basins are data and computation intensive because of the combined nature of hydrodynamics, complex forcings and heterogeneous parameter fields. Application of these models at fine temporal and spatial resolutions, and on large problem domains, is facilitated by parallel computation on multi-processor clusters. Notably, the computation efficiency of parallel simulations is crucially determined by the efficiency with which data are divided-and-distributed in a multi-processor environment and how the information is shared between processors. While numerous data partitioning algorithms exist and have been extensively studied in computer science literature, detailed elucidation of the role of hydrologic model structure on data partitioning has not been presented yet. In addition, the relative role of computational load balance and interprocessor communication on parallel computation efficiency of a hydrologic model is not known. Considering the unstructured domain discretization scheme used in PIHM hydrologic model as an example, the paper first presents a generic methodology for incorporating hydrologic factors in optimal domain partitioning algorithms. The partitions are then used to explore the isolated role of computation load balance and interprocessor communication on parallel efficiency. Results confirm that parallel simulations on partitions that minimize interprocessor communication and divide the computational load equally are the most efficient. More importantly, load balance between processors is observed to be a more sensitive control on parallel efficiency than minimization of interprocessor communication. Further analyses of the efficiency and scalability of the parallel code for different partitioning configurations reveal a direct correspondence between parallel efficiency and theoretical metrics such as load balance ratio and communication to computation ratio. Results indicate that theoretical metrics can be used for the selection of best partitions before computationally intensive parallel simulations are performed. The study serves as a proof-of-concept evaluation of the impact of computation and communication on the efficiency of parallelized distributed hydrologic models at multiple resolutions.

Keywords

PIHM; Integrated modeling; Parallel computing; Graph theory; Surface water; Ground water

***Corresponding author:** Mukesh Kumar, Assistant Professor of Hydrology and Water Resources, Nicholas School of the Environment, Duke University, Durham, NC 27708-0328, U.S.A., Tel: +1-919-681-7837; E-mail: mukesh.kumar@duke.edu

Received: May 16, 2014 **Accepted:** January 19, 2015 **Published:** January 23, 2015

Introduction

Physics-based distributed hydrologic models (DHMs) simulate multiple hydrologic states at numerous locations within a watershed, and at fine time intervals [1-7]. As the spatio-temporal resolution of model simulation becomes finer and number of predicted hydrologic states becomes larger, model simulations become more and more computationally intensive, rendering solution of large problems intractable or at least not suitable for near real-time predictions on a serial computer. The computational burden is further enhanced by the use of finer resolution (≤ 30m) geospatial products that are inputs to these models. The need to run DHMs at fine spatio-temporal resolution for multiple states/parameters over large domains necessitates the use of high performance computing (HPC) systems.

Growth in computing power and speed along with rapid decrease in hardware costs [8] have facilitated the use of HPC systems in a wide variety of science and engineering applications, including for DHM simulations. One of the earlier implementations of a parallel DHM was performed by Morton et al. [9] who developed a parallelized code for simulating hydrologic processes in Arctic regions, primarily on Cray architectures. Performance gains were found for 8 and 32 processors using MPI and CRAFT implementations respectively. Apostolopoulos and Georgakakos [10] used ENCORE Parallel FORTRAN for parallelization of a sub-basin based semi-distributed hydrologic model on 14 processors. Both of these implementations were limited to parallelization of surface flow network. Cui et al. [11] parallelized DHMs by partitioning the watershed into sub-basins, where they specifically noted the adverse effect on parallelization efficiency due to imbalance in computational load assignment on different processors. Cui et al. [11] tackled the problem of load imbalance by redistribution of load between processors using sending-by-pairs, circular-send and sending-by-percentage methodology. The technique transmitted data from overloaded to underloaded processors in order to balance load. However, the methodology greatly increased the communication between processors. Cheng et al. and Vivoni et al. [12,13] alleviated these problems by optimal partitioning of the model domain using METIS [14] software, which ensured both the load balance and minimization of interprocessor communication. pWASH123D model code [12] demonstrated performance gains up till 32 processors. Vivoni et al. [13] reported a maximum speed up of 70 (efficiency of ~ 0.14) for 512 processors. Kollet and Maxwell [6] presented a parallelized, structured-grid based, fully coupled model by coupling overland flow with PARFLOW groundwater flow model [15]. Application of the parallelized coupled model on an experimental setup to study surface-subsurface flow interactions recorded an efficiency of 0.82 for 100 processors. Kollet et al. [16] demonstrated a parallel efficiency of around 0.6 for an application of parallel PARFLOW model on 16384 processors, and made a compelling case to finally be able to perform large-scale fully-coupled hydrologic simulations at fine resolutions within reasonable time. The varied computational efficiency of different parallelization efforts is partially attributable to two factors: efficiency with which the modeling problem is divided-and-distributed in a multi-processor environment and the amount of information that is shared between processors. However, there has been no proof-of-concept evaluation and demonstration of the impact of these two factors on the

efficiency of parallelized distributed hydrologic models. In addition, the relative influence of partitioning of the computational load and information shared between processors on computation efficiency has not been presented before for DHMs. Finally, details about how the characteristics of DHMs can be incorporated in existing optimal partitioning strategies are yet to be delineated.

In this paper, parallel implementation of a fully-coupled DHM called Pennstate Integrated Hydrologic Model (PIHM) for simulating land-surface-subsurface hydrologic states is presented on a cluster of 512 processors. First, we present a generic methodology for incorporating hydrologic factors, such as the number of hydrologic processes, coupling between processes and flow network topology, in optimal domain partitioning algorithms. The model is then used to evaluate how partitioning of model domain influences the computation efficiency of parallel DHMs in relation to computational load balance and inter-processor communication across processors. Next, the relative role of load balance between processors and minimization of interprocessor communication on parallel efficiency is presented. Finally, relations between theoretical and actual metrics of parallel efficiency are evaluated to explore if theoretical metrics can be used for comparing parallel efficiency of different partitions.

PIHM Formulation

PIHM is a fully coupled, physically-based, spatially distributed hydrologic model. It simulates six hydrologic states (canopy interception storage, snow water equivalent, 2-D overland flow, soil moisture in the surface layer, unsaturated zone soil moisture and groundwater levels) on each unstructured element of the watershed using a semi-discrete finite-volume approach [4,5,17]. Streamflow simulation in the model is based on a depth-averaged 1-D diffusive wave equation, surface flow uses a depth-averaged 2-D diffusive wave approximation of the Saint Venant equations, and subsurface flow is based on depth-averaged, moving boundary approximation of Richards's equation [17]. The partial differential equations (PDEs) governing each of these processes are locally reduced to ordinary differential equations (ODEs) by integration on a spatial unit element. The generic semi-discrete form of ODE that defines all the hydrologic processes incorporated in PIHM is represented as

$$A_i \frac{d\bar{\psi}}{dt} = \sum_j \vec{n}.\vec{G}_j A_{ij} + \sum_k \vec{n}.\vec{F}_k A_{ik} + \bar{S}_\psi V_i \quad (1)$$

Where $\bar{\psi}(L)$ is average volumetric conservative scalar per unit planimetric control volume area A_i, $\bar{S}_\psi$ is average source/sink rate per unit control volume, $\vec{G}$ and $\vec{F}$ are vertical and lateral flux terms respectively, and $\vec{n}$ is normal vector to the surface *j* of the control volume *i*. Relevant ODEs for all the hydrologic processes in the PIHM control volume are shown in Table 1. For more details about the individual process equations, readers are referred to [17]. The model has been successfully applied at multiple scales and in diverse hydro-climatological settings in both North America and Europe [18-21].

Parallelization Strategy

Here, we use a domain partitioning approach for parallelizing PIHM. The methodology, also referred as Single Program Multiple Data technique [22], entails running the same model code concurrently on multiple processors. Each processor is assigned its own part of the data or ODEs to work on. In order to preserve spatial coupling of processes (or lateral fluxes, *f*[] in Table 1) between control volumes that lie on different processors, appropriate communication of hydrologic states between processors is implemented to ensure that the solution of a serial and parallel problem is exactly the same. The methodology is potentially scalable and can be used to simulate the parallel model on thousands of processors. However, its effectiveness depends on how the data are divided-and-distributed in a multi-

Table 1: Definition of coupling function and the lateral and vertical fluxes across the faces of a control volume. *i* and *j* are indices of neighboring control volumes. ‖ ‖ denotes conditional terms for grids that are neighbors of a river element. Explanation of symbols is in Appendix.

Prismatic Element					
Control Volume	$\bar{\psi}$ (State)	$\vec{G}$ (Vertical Flux)	$\vec{F}$ (Lateral Flux)	$\bar{S}_\psi$ (Source/ Sink)	*f*[] (Coupling Flux Function)
Interception	ψ_0	$\vec{G}_3 - \vec{G}_4 - \vec{G}_5$	--	--	$\vec{G}_5 \equiv f[\psi_0]$
Snow	ψ_1	$\vec{G}_3 - \vec{G}_6$	--	--	$\vec{G}_6 \equiv f[\psi_1]$
Surface Flow	ψ_2	$\vec{G}_3 - \vec{G}_0 - \vec{G}_7 + \vec{G}_5 + \vec{G}_6$	$\vec{F}_0 + \lVert\vec{F}_1\rVert$	--	$\vec{F}_0 \equiv f[\psi_{2i},\psi_{2j}], \vec{G}_0 \equiv f[\psi_2,\psi_3]$ $\vec{G}_5 \equiv f[\psi_0], \vec{G}_6 \equiv f[\psi_1], \lVert\vec{F}_1\rVert \equiv f[\psi_5,\psi_2]$
Unsaturated Zone	ψ_3	$\vec{G}_0 - \vec{G}_1 - \vec{G}_8 - \vec{G}_9$	--	--	$\vec{G}_0 \equiv f[\psi_2,\psi_3], \vec{G}_1 \equiv f[\psi_3,\psi_4]$ $\vec{G}_8 \equiv f[\psi_3], \vec{G}_9 \equiv f[\psi_3,\psi_0]$
Saturated Zone	ψ_4	$\vec{G}_1$	$\vec{F}_2 + \lVert\vec{F}_3\rVert + \lVert\vec{F}_4\rVert$	$-S_1$	$\vec{G}_1 \equiv f[\psi_3,\psi_4], \vec{F}_2 \equiv f[\psi_{4i},\psi_{4j}]$ $\lVert\vec{F}_3\rVert \equiv f[\psi_5,\psi_4], \lVert\vec{F}_4\rVert \equiv f[\psi_6,\psi_4]$
Linear Element					
Channel zone	ψ_5	$\vec{G}_3 - \vec{G}_2 - \vec{G}_7$	$\lVert\vec{F}_3\rVert + \vec{F}_5 + \lVert\vec{F}_1\rVert$	--	$\vec{G}_2 \equiv f[\psi_5,\psi_6], \vec{F}_5 \equiv f[\psi_{5i},\psi_{5j}]$ $\lVert\vec{F}_1\rVert \equiv f[\psi_5,\psi_2], \lVert\vec{F}_3\rVert \equiv f[\psi_5,\psi_4]$
Sub-Channel Zone	ψ_6	$\vec{G}_2$	$\lVert\vec{F}_4\rVert$	--	$\lVert\vec{F}_4\rVert \equiv f[\psi_6,\psi_4], \vec{G}_2 \equiv f[\psi_5,\psi_6]$

processor environment and the efficiency with which information is shared between processors.

Domain partitioning

Efficient data partitioning involves satisfying two objectives: load balancing and minimization of interprocessor communication. Load balancing ensures that computation on each processor finishes simultaneously during each time step, thus avoiding any idle-time delay incurred on processors that finish their jobs earlier than others. This leads to the most efficient use of existing parallel computing resources. The second objective is to minimize communication between processors. Although the number of communication operations is generally much less than computation, the cost of accessing memory on other processors is relatively much more time-expensive with respect to accessing variables from a local processor. The cost of processor communication is proportional to amount of data shared, frequency of data sharing, and latency and bandwidth of interconnection network. One obvious strategy to minimize communication is by assigning computation to only a small number of processors thus reducing the need to communicate with other processors in a cluster. However, that will lead to load imbalance since few processors would do all the work while the others will remain idle. An optimal partition should ensure both load balance and minimal communication.

Graph definition and weight assignment: Consideration of hydrologic factors: A partitioning problem is generally formulated on an undirected communication graph [23], which is obtained by connecting unit elements that exchange information. For the cell-centered finite volume formulation in PIHM, vertices of the communication graph are the centroids of discretization elements and edge of the graph is the connecting segment joining the vertices (Figure 1). Edges connect the neighboring elements that share information during a time step. Such a graph is called the *dual graph* of a mesh. The edge and vertex weight of a dual graph represents the computational load at each element and the communication between neighboring elements respectively [23].

Weights on a dual graph of a DHM simulation domain can be assigned based on hydrologic factors such as the number of hydrologic processes solved at each discretization element, flow topology, and level of spatial coupling between processes. An increase in number of prognosticated states proportionally increases the amount of computation and the data shared with neighboring elements. This directly impacts computation and inter-processor communication time, and hence also the vertex and edge weights. The flow topology also determines the communication volume. Depending on if a model calculates overland flow flux in the direction of maximum elevation gradient or if it accounts for head magnitudes in all the neighboring elements of a cell, communication volume may be different. For example, to evaluate overland flux in single flow direction based models (e.g. tRIBS), dual mesh edges corresponding to unstructured mesh faces that do not interact with their neighbors should have an edge weight of zero. In contrast, positive edge weights will have to be assigned to all edges of a dual mesh in multiple flow direction based models (e.g. ModHMS, PIHM). Level of spatial coupling, i.e. the number of hydrologic processes that interact laterally with neighboring cells, also affects communication volume and hence the edge weight. For example, models such as FIHM/ PIHM3D [5] simulate lateral flow in both unsaturated and saturated zone whereas PIHM only considers sub-surface lateral flux between groundwater states. So edge weights in FIHM have to be larger than in PIHM to account for larger communication. It is to be noted that in addition to hydrologic factors, design and configuration of a parallel computing cluster such as the relative speed of individual processor nodes, and latency, bandwidth, diameter and degree of interconnect, can also contribute to heterogeneous computing and communication between processors, and hence should be duly accounted for in assignment of dual mesh weights.

Weight assignment on a PIHM dual mesh is unusual even among unstructured grid based models. In addition to communication between triangular mesh watershed elements, communication in PIHM is also defined between river and watershed elements (triangles) and between upstream and downstream river elements (Figure 1). Computation weights are assigned to vertices of a dual graph in proportion to the number of prediction variables or the number of ODEs that are solved on a discretized element. So a computation weight of 6 is assigned to each triangular element, as six hydrologic states are solved on each of them. On linear river elements, PIHM solves for two states (stream stage and groundwater depth below the river). Similar to the assignment of computation weight on vertices of the dual graph, edges of dual graphs are assigned weights in proportion to number of laterally coupled physical states that are shared between neighboring linear or triangular elements. In PIHM, two neighboring elements (e.g. triangle-triangle, river-triangle or river-river) share information about groundwater depths and overland flow depths in order to calculate the lateral fluxes at each time step. Hence, a communication weight of two is assigned to all the dual graph edges. We note that computation and communication weights on dual meshes can be similarly implemented for other fully-distributed models that are based on finite difference, finite element and finite volume methods. The magnitude of computation and communication weights, however, will differ depending on the number of predicted states on each unit element, number of spatially coupled processes, shape of each unit element and flow topology between neighboring elements.

Domain partitioning algorithms: Using a dual graph with n weighted vertices and m weighted edges, the objective is then to divide the vertices into P partition sets in such a way that the sum of vertex weights in each set is as close as possible and sum of the weights of edges crossing between sets is minimized. The posed problem is NP-complete [24] and so it is hard to obtain globally optimal solutions. Several near-optimal techniques such as the K-L algorithm [25],

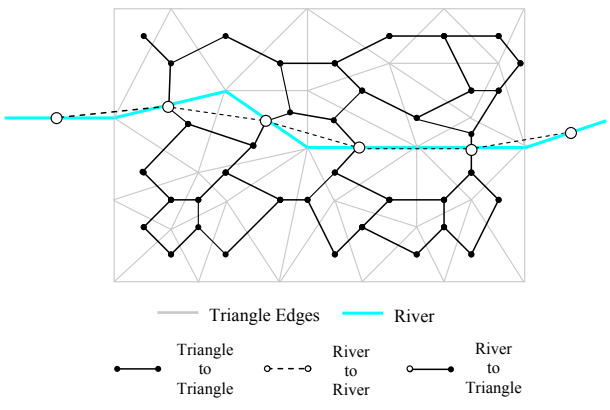

Figure 1: Dual graph for a representative unstructured discretization of PIHM domain.

simulated annealing [26], genetic algorithm [27], greedy method [28], recursive inertial bisection (RIB) [29], recursive graph bisection (RGB) [30], recursive spectral bisection (RSB) [31,32], multilevel partitioning (ML) [33], and hybrid methods such as multilevel-KL partitioning (ML-KL) [34] and RSB and RGB algorithms refined by KL method (RSB-KL, RGB-KL) [35] exist for partitioning the dual graph. RSB algorithms have been previously found to outperform or serve as a better compromise to RGB, RIB, greedy and simulated annealing methods [30,32,36]. Readers are encouraged to look into respective references to see details of the algorithms. Implementations of these algorithms are available in numerous state-of-the-art partitioning software packages including CHACO [37], JOSTLE [38], METIS [14] and RALPAR [35]. In this paper, the best partition from among RGB, RSB, RGB-KL, ML-KL and RSB-KL is used for further analyses. The best partition is selected based on theoretical measures of load balance and communication minimization. Load balance of a partition-set is measured using load balance ratio (*LBR*), which is calculated as

$$LBR = \frac{\sum_{i}^{N_p} N_{o_i}}{N_p \max(N_{o_i})} \times 100 \quad (2)$$

Where N_p is number of processors and N_{o_i} is number of dual mesh nodes in i^{th} partition. For a perfectly balanced partition, the maximum partition size and average size of each partition is all the same. So *LBR* is equal to 100 for this case. With increase in load imbalance, numerator in Equation 2 remains the same, while denominator increases. Hence *LBR* is less than 100 for an imbalanced load case. Communication volume, which is quantified as edge-cut (E_c), is equal to the sum of weights of dual-mesh edges that connect vertices from different partitions. RGB and RGB-KL partitions were generated using RALPAR while RSB, RSB-KL and ML-KL partitions were obtained using CHACO software. We note that the partitions obtained in this analysis are static, and are used throughout the model simulation. In simulations where compute times on processors are markedly different, static partitioning may not be the most efficient strategy.

PIHM code parallelization

The governing ODEs for all the considered physical processes are solved using an implicit Newton-Krylov based solver called CVODE [39]. Within each integrator time step, the rate of change in state variables (right hand side of ODEs in Equation 1) on each partition (and hence on separate processors) is evaluated. The Message Passing Interface (MPI) system [40] is used to carry out parallel communications between partitions at synchronization points, thus ensuring that data from other processes are available locally when needed. Two sets of send and receive operations are executed. The first communication-set exchanges information about hydrologic states between processors. This is executed at the beginning of the subroutine that defines process ODEs in parallel-PIHM code. The second communication-set is executed from within the solver at the end of ODE evaluation subroutine. This communication-set includes MPI global reduction operations such as dot products, weighted root-mean-square norms and linear sums. These operations are launched at each convergence iteration step to check for the absolute and relative tolerance metric criteria [41]. This location in the code also acts as the synchronization point, as quantification of fluxes is a pre-requisite to any global convergence criteria evaluations. A successful convergence at any iteration step indicates fulfillment of tolerance criteria for all ODEs in the model domain, instead of for a local set of ODEs on a particular partition. We note that in this step, overlap between computation and communication operations that can potentially reduce the wall clock time is not executed, and will be addressed in future versions of the code. The developed parallelized code is run on IBM x3450 1U Rackmount Server with 64 GB of ECC RAM and Dual 3.0 GHz Intel Xeon E5472 (Woodcrest) Quad-Core Processors. The cluster is a shared resource with limited number of users (less than ten), and consisted of 756 active processors.

Experiments, Results and Discussions

To demonstrate the effectiveness and scalability of the parallel model in relation to how a domain is partitioned, four representative partitioning configurations are considered: a) an optimal partition where computational load is balanced and communication is minimum, b) a partition with balanced load but with large interprocessor communication, c) a partition with minimum interprocessor communication but with imbalanced load, and d) a partition with both load imbalance and large interprocessor communication. The results of four configurations are compared based on the parallel efficiency metric [42], *E*, which is quantified as the ratio of Speedup (S_{N_P}) to the number of processors (N_p), and is given by

$$E = \frac{S_{N_P}}{N_P} \quad (3)$$

Speedup, S_{N_P}, is defined as the ratio of wall-clock time for a serial program to the time for a parallel version of the same program. Scalability comparisons between the four configurations involve evaluation of respective parallel implementation's capability to demonstrate a proportionate increase in speedup with additional processors.

The parallel model is implemented for the Little Juniata River Watershed, located in south central Pennsylvania. The watershed size is 845.6 sq. km, and is characterized by significant complexity in the bedrock geology, soil, land cover, precipitation and elevation [17]. All physiographic and hydroclimatic data and other topological relations needed to perform model simulations are automatically mapped to the model unstructured mesh using PIHMgis [43]. Constrained Delaunay triangulation is used to discretize the domain at four spatial resolutions (Table 2): 412, 243, 169 and 99 m. Constraints include hydrographic features such as rivers and sub-watershed boundaries, and thematic features such as soil and land cover classes [44]. Case I discretization (Table 2) is obtained by only using hydrographic features as constraints. Next two discretization levels are generated by incrementally considering land cover and soil boundaries, along with hydrographic boundaries as constraints. Finest discretization (case IV) is generated by setting maximum-triangle-area constraint of 0.016 sq. km in Triangle [45], in addition to internal boundary constraints used in case III. The problem size for four discretization levels range from 32566 to 534048 ODEs. Parallel simulations are performed for the two month period (November 1 to December 30, 1983) at each discretization resolution. This standardizes comparisons and discounts the effect of change in hydrologic process dynamics between model runs. We note that even though model simulations are being performed for the same duration in all four discretization configurations, changes in resolution does lead to differences in representation of topography and hydro-geologic properties and hence also in the hydrologic interactions. This indicates that the model simulations at different resolutions are not identical and may have

differences in process dynamics. The simulation period comprises of 30 precipitation events with a total precipitation of 218 mm (average precipitation = 3.7 mm/d). The precipitation duration during the two-month period was about 22 days, out of which snow events spanned 7 days. The serial code took 273 hours to finish 2 months of simulation on case IV discretization. Choice of the finest discretization resolution and length of simulation period were ceiled by the time it takes to solve the largest problem on a single processor, as it has to be less than the maximum contiguous compute time available on the cluster. It is to be noted that maximal range of discretization resolutions and processor (partition) size considered in each experiment is governed by the need to optimally utilize limited wall-clock times available to each individual user on shared computing cluster, while still being able to sufficiently confirm the goals of each experiment. The simulation settings and hydrologic response predictions of the model (on one processor) for a two year model run are presented in Kumar [17]. Here we focus only on the computational aspect of the model simulation.

Parallel model efficiency of an optimal domain partition: Balanced load and minimum interprocessor communication

The first experiment uses load-balanced partitions with minimum inter-processor communication. Partitioning is performed for all four discretization levels (Table 2) and the parallel efficiency is evaluated for 2, 4, 8, 16, 32, 64, 128, 256 and 512 processors (Figure 2).

Figure 2 shows results of only the best theoretical partition configuration from among RGB, RSB, RGB-KL, ML-KL and RSB-KL at each scale. The best partition was selected based on *LBR* and E_c metrics (see Section "Parallel model efficiency of an optimal domain partition: Balanced load and minimum interprocessor communication"). Notably, all of the five algorithms considered here ensure an identical *LBR* of close to 100%, but they differ in the number of edge-cuts i.e. in their effectiveness to minimize communication. For all four discretization levels, either RSB-KL or ML-KL partition outperformed other algorithms considered in this study in terms of minimization of E_c (Table 3). The experiment suggests that for larger number of partitions, ML-KL generally yields the best partition (Figure 2, Table 3). Notably, the spatial distribution of partitions obtained from these algorithms look very similar across different discretizations. For example, partitions on smaller number of processors share their boundaries with partitions on larger number of processors. This is because of the recursive nature of partitioning algorithms, which subdivide the coarser partitions to obtain finer ones. The spatial distribution of partitions obtained using RSB-KL and ML-KL algorithms, however, can be fairly different. It is to be noted that partition boundaries for river and triangles coincide with each other as this leads to reduction in overall communication.

For the same level of discretization (problem size), efficiency of the parallel model decreases with increase in the number of processors (Figure 3). For case I, which corresponds to the coarsest discretization level and hence the smallest problem size, efficiency reduces from 1 to 0.087 as the number of processors increase from 1 to 512. Even though running at a fairly low efficiency, the parallel model runs around 73 and 45 times faster than the serial code on 256 and 512 processors respectively. The rate of decrease in efficiency with increase in the number of processors is much smaller for larger problem sizes. For example, parallel efficiency in case IV is close to one for up until 128 processors. Even for 512 processors, the efficiency in case IV reduces only to 0.6 and is much larger than the efficiency in cases I, II and III (Figure 3). The parallel code runs around 312 times faster on 512 processors (for case IV) with respect to the serial version of the model code. Comparatively the computation is 185, 103 and 45 times faster on 512 processors for cases III, II and I respectively. We note that for some cases, efficiency is slightly larger than one (Figure 3). This is attributable to: a) a large working-set size (amount of memory needed to compute the problem) relative to available memory on individual processors, when data are mapped onto small number of processors. As a result, data are fetched from the disk, a relatively slower option than data fetch from memory. In contrast, the partitioned data size and hence the working-set is smaller for large number of processors. This results in a faster data fetch from memory, leading to super-linear speedups; b) pre-fetching of data in a buffer, which leads to faster access of data by other cores (in a quad-core processor),in addition to the one that is fetching the data. This saves time as shared cores do not have to individually fetch data, and can access it directly from the buffer, thus contributing to higher efficiency. Mild sensitivity of the efficiency results to unsustainable steady throughput from processors cannot be discounted either.

The decrease in efficiency with decrease in problem size and increase in processors can be explained by rewriting Equation 3 as

$$E = \frac{T_1 / T_{N_P}}{N_P} = \frac{T_{ser} + T_{par}}{N_P\left(T_{ser} + \frac{T_{par}}{N_P} + T_{comm}\right)} \approx \frac{1}{1 + N_P R} \quad (4)$$

Where T_{ser} and T_{par} are wall clock times for the serial and parallelizable part of the code on a single processor, T_{comm} is the inter-processor communication time, and R is the ratio of communication to computation, $\frac{T_{comm}}{T_{par}}$. Equation 4 is derived by considering T_{ser} « T_{par}, a reasonable assumption as parallelizable part of the code is the time intensive part where all time-evolving computations are performed. Since T_{comm} is directly proportional to E_c (see Section "Parallel model efficiency of an optimal domain partition: Balanced load and minimum interprocessor communication" for details), and T_{par} is directly proportional to $No\left(=\sum No_i\right)$, Equation 4 can be

Table 2: The four levels of domain discretization of Little Juniata Watershed. Bdd. stands for boundary.

Discretization	Number of triangular elements (NTE)	Number of river elements (NRE)	Total number of ODEs (= 6* NTE +2*NRE)	Minimum, Maximum, Mean Area of Triangular Element (sq. m)	Minimum, Maximum, Mean Length of River Element (m)	Constraints
Case I	5065	1088	32566	1163, 1323137, 169607	38, 1725, 549	Sub-watershed Bdd. +Rivers
Case II	14553	1751	90820	5.4, 149872, 59029	2.7, 926, 341	Sub-watershed Bdd. +Rivers + Land Cover
Case III	30155	2608	186146	0.03, 59985, 28488	0.3, 564, 229	Sub-watershed Bdd. +Rivers + Land Cover + Soil
Case IV	87645	4089	534048	0.03, 15999, 9801	0.3, 328, 146	Sub-watershed Bdd. +Rivers + Land Cover + Soil

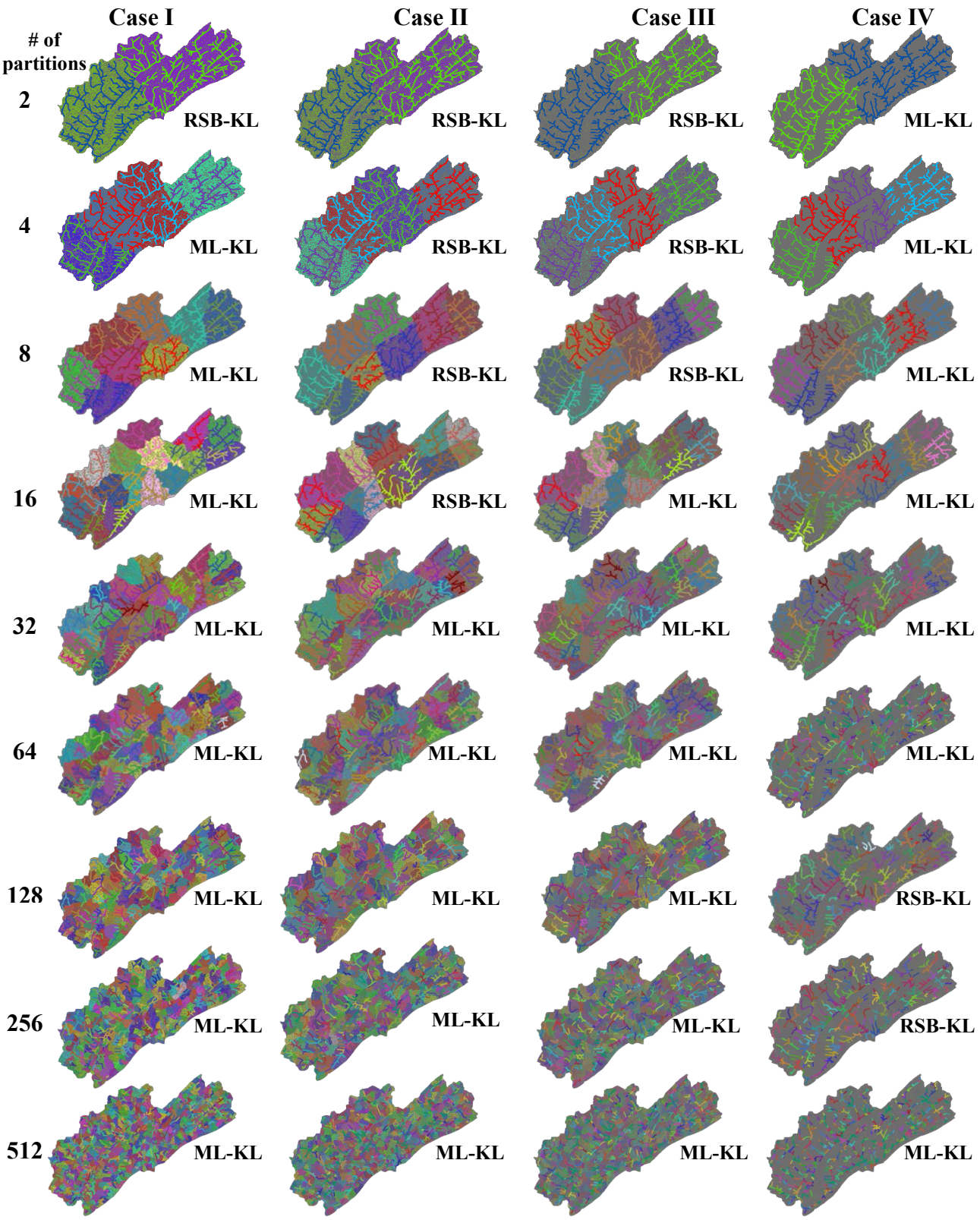

Figure 2: Optimal domain partitions for four discretization levels (Table 2) in the Little Juniata watershed on 2, 4, 8, 16, 32, 64, 128, 256 and 512 processors. Partitioning algorithm that produces the best theoretical efficiency among RGB, RSB, RGB-KL, RSB-KL, ML-KL is listed corresponding to each partition.

Table 3: Number of edge-cuts in partition. Numbers in shaded boxes denote the smallest edge-cut for a given discretization resolution and partition (processor) size

Discretization Level →	Case I					Case II					Case III					Case IV				
Number of Processors →	RGB	RSB	RGB-KL	RSB-KL	ML-KL	RGB	RSB	RGB-KL	RSB-KL	ML-KL	RGB	RSB	RGB-KL	RSB-KL	ML-KL	RGB	RSB	RGB-KL	RSB-KL	ML-KL
2	94	90	96	84	88	132	126	130	120	124	210	196	202	194	198	352	334	344	326	324
4	220	208	216	202	200	326	312	324	296	300	484	464	482	450	452	852	830	856	816	808
8	448	420	442	408	408	716	668	704	640	646	946	900	922	890	892	1698	1644	1704	1602	1592
16	774	710	740	696	692	1214	1136	1192	1112	1110	1648	1580	1640	1534	1532	2968	2866	2964	2800	2798
32	1170	1128	1162	1112	1098	1878	1774	1828	1748	1738	2538	2418	2456	2400	2396	4668	4412	4602	4308	4298
64	1840	1718	1810	1674	1674	2788	2646	2714	2594	2582	3926	3758	3854	3684	3676	6812	6686	6978	6536	6518
128	2780	2664	2764	2588	2588	4158	3874	4016	3820	3820	5820	5578	5794	5420	5414	10364	9702	10056	9504	9496
256	4210	3976	4172	3878	3864	6170	5872	6170	5728	5728	8608	8234	8450	8078	8064	14638	13964	14474	13676	13670
512	6222	5952	6142	5826	5808	9230	8788	9156	8608	8604	12694	12140	12458	11910	11902	20624	19836	20134	19688	19670

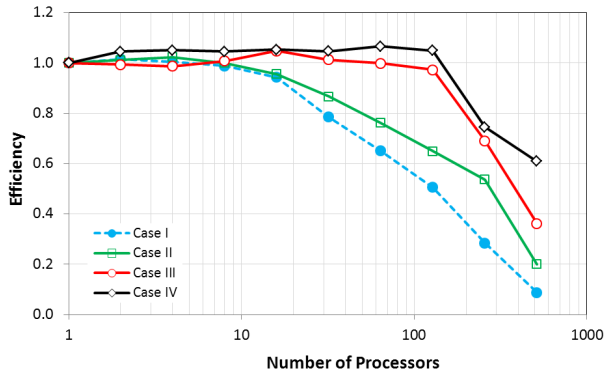

Figure 3: Efficiency of parallel PIHM for four discretization levels (see Table 2 for details) on 1, 2, 4, 8, 16, 32, 64, 128, 256 and 512 processors.

rewritten as:

$$E = \frac{1}{1 + N_P R} = \frac{1}{1 + \alpha N_P \frac{E_c}{N_o}} \quad (5)$$

Where α is a proportionality constant. For a given problem size (constant N_o), increase in number of partitions (N_p) results in proportional increase in E_c. This translates to increase in R (Table 4), and hence a decrease in E (based on Equation 5), with increase in N_P. The results point to reduced efficiency on larger number of processors because of an increase in the amount of communication between processors. For identical N_p but larger problem sizes, both E_c and N_o increases. However, increase in N_o is much larger than the increase in E_c. This leads to a decrease in R (Table 4) and hence an increase in E for finer discretizations (based on Equation 5).

Larger efficiency of the parallel PIHM model for finer discretizations of the model domaindemonstrates the potential of applying the parallel model to even larger problem sizes and number of processors (~1000s).

Parallel model efficiency of sub-optimal domain partitions

The effectiveness of optimal domain partitioning with respect to other sub-optimal partitioning configurations is explored next. In this context, three experiments are conducted. The first two configurations highlight the isolated impacts of load balance and interprocessor communication on parallel efficiency, while the third configuration considers their combined impact.

Balanced computation load, but inefficient interprocessor communication: In the first configuration, partitions are generated such that computation load across processors is balanced, but without accounting for minimization of communication between processors. In this regard, two experiments are conducted. First, the partitions are generated by randomly assigning discretized cells and hence the ODEs to different processors while ensuring that the total number of cells in each partition is the same (Figure 4). The method yields partitions with number of edge-cuts being almost an order of magnitude higher than in the optimal partitioning case (discussed in section "Parallel model efficiency of an optimal domain partition: Balanced load and minimum interprocessor communication"). Using case I as a representative discretization in this experiment, the

Table 4: Ratio of edge cut (E_c) and problem size (N_o), a surrogate for communication to computation ratio (R), for the four levels of discretization on 2, 4, 8, 16, 32, 64, 128, 256 and 512 processors.

Discretization Level (Across) / Number of Processors (Down)	Case I	Case II	Case III	Case IV
2	0.003	0.001	0.001	0.001
4	0.006	0.003	0.002	0.002
8	0.013	0.007	0.005	0.003
16	0.021	0.012	0.008	0.005
32	0.034	0.019	0.013	0.008
64	0.051	0.028	0.020	0.012
128	0.079	0.042	0.029	0.018
256	0.119	0.063	0.043	0.026
512	0.178	0.095	0.064	0.037

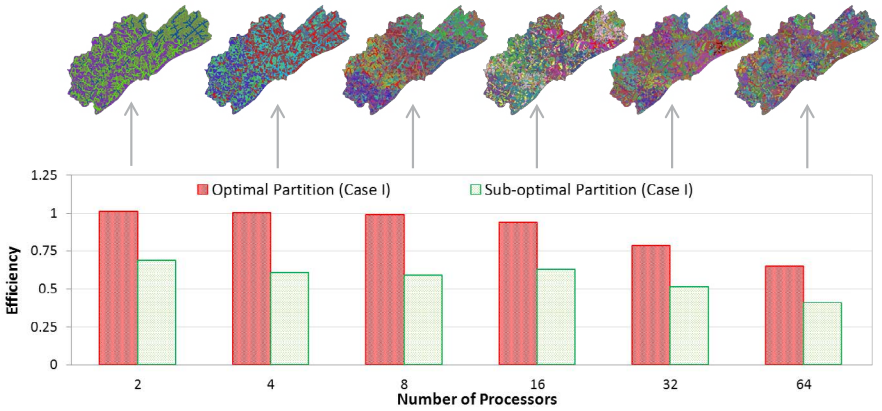

Figure 4: Comparison of parallel efficiency between optimal partitions and partitions with load balance but inefficient communications, using Case I discretization.

number of edge-cuts for the optimal partitioning case and for this case on 2, 4, 8, 16, 32 and 64 processors are (82, 5690), (196, 8354), (406, 10688), (682, 11634), (1082, 12318) and (1654, 13102) respectively. For the same problem size, since edge-cuts for optimal partitioning case is always smaller than that for load balanced partitions with efficient interprocessor communication, following Equation 5 they over-perform in terms of parallel efficiency for all processor sets (Figure 4). A consistently lower parallel efficiency of these randomly distributed non-contiguous partitions and their explain ability based on communication to computation ratio (R) underscores that the results are not chance agreements. Next, the sensitivity of parallel efficiency to edge cuts (or communication volume) is further explored by generating four additional partitions with increasing number of edge-cuts. Starting with an optimal partition configuration on 64 processors, same numbers of triangles are randomly selected from two different partitions and are then assigned to the other partition. This ensures load balance, but increases edge-cuts. The process is repeated for additional pairs of partitions to obtain varying number of edge-cuts. Edge-cut and corresponding parallel efficiency for the five partition sets on 64 processors are (1654, 0.65), (2974, 0.58), (5872, 0.53), (10942, 0.48) and (13102, 0.41) respectively. Results suggest that for load-balanced partitions, increase in the number of edge-cuts result in reduction of parallel efficiency, when the problem size and number of processors are kept constant. The two experiments highlight the role of minimization of edge-cuts during domain partitioning to obtain best parallel efficiency.

Minimum interprocessor communication, but imbalanced computation load: To evaluate the impact of computational load balance on parallel efficiency of the model, an alternate sub-optimal configuration is considered where the load is imbalanced but minimum communication between processors is maintained. This partition configuration is obtained by simple reassignment of cells of an optimal partition (derived in section "Parallel model efficiency of an optimal domain partition: Balanced load and minimum interprocessor communication"). The mapping of cells for (N_p-2) partitions is left as is. For the remaining two partitions, cells from one are assigned to the other (Figure 5), thus altering load balance. For a perfectly balanced partition set, the *LBR* is equal to 100. As the load imbalance increases, the *LBR* becomes smaller. For case I discretization on 16 processors, the simulation time with respect to a load-balanced case, increases with increasing load imbalance (decreasing *LBR*). A consistent under-performance of the representative load imbalanced configuration and the decrease in its efficiency (increasing wall clock time) suggests that similar trend in computation efficiency can be expected for partition configurations with smaller *LBRs* but with near-identical communication volume. It is to be noted that a reduction in *LBR* from around 100 % to 55 % lead to a decrease in parallel computation efficiency from 0.94 to 0.4 on 16 processors. In contrast, 16 times increase in edge cuts (a surrogate for communication volume) resulted in computation efficiency to only reduce from 0.94 to 0.6 (See Section "Balanced computation load, but inefficient interprocessor communication").

Imbalanced load and inefficient interprocessor communication: A sub-watershed based partitioning configuration, obtained by

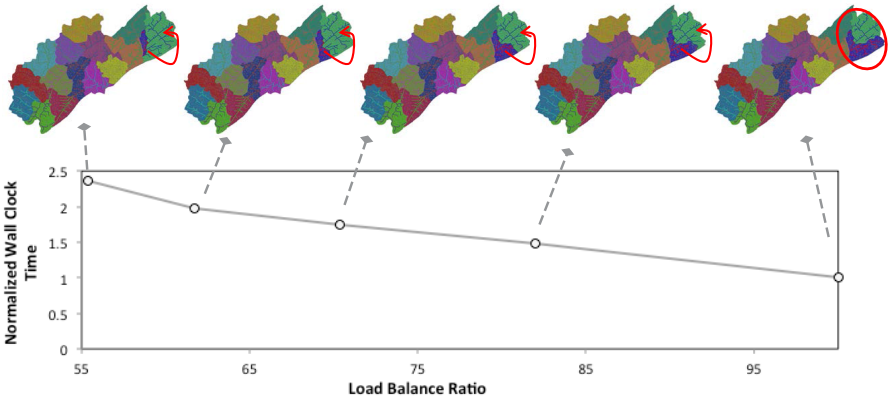

Figure 5: Comparison of wall clock time for partitions with different load balance ratios, on a Case II discretization mapped to 16 processors using RSB-KL algorithm. The normalized wall clock time represents the ratio of wall clock time with respect to the case where load balance ratio is equal to 100%. The corresponding load imbalanced partitions are also shown (top). The load imbalance is simulated by assigning discretization elements from one partition to another. Red ellipse highlights the location of participating partitions. Re-assignment of unit elements is indicated by the arrow

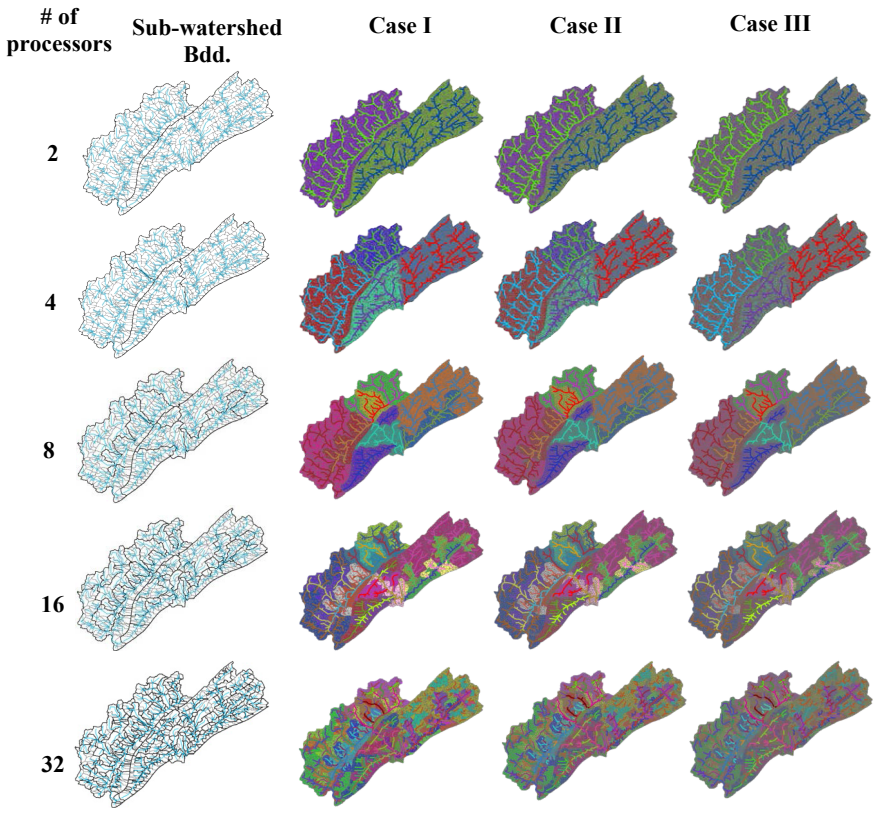

Figure 6: Sub-watershed based partitioning of the model domain on 2, 4, 8, 16 and 32 processors. Sub-watersheds are shown in the left-most column in light grey. Dark grey internal boundaries depict partition boundaries

mapping the sub-watersheds (Figure 6) of a fifth Strahler-order stream network to different processors, is considered next. First, the sub-watersheds are aggregated into two contiguous units such that both units drain into a fifth order stream. The two aggregates are then mapped to different processors. For mapping on four processors, sub-watersheds are aggregated into four contiguous units in such a way that each unit drains into fourth or higher order streams. The process is repeated for mapping on larger number of processors. Partitions are obtained for 2, 4, 8, 16 and 32 processors for three discretization resolutions (cases I, II and III, Figure 6). Sub-watershed based partitions generally have a smaller *LBR* and larger *R* with respect to optimal partitions (Table 5). The results show that parallel efficiency, even for the largest problem (case III) in this case, is consistently smaller than the optimal partitioning case (Figure 7). Parallel

Table 5: Comparison of load balance ratio (*LBR*) and the ratio of edge cut (E_c) and problem size (N_o), a surrogate for communication to computation ratio (*R*), between optimal partitions (OP) and sub-watershed based partitions (SWP)

Metric →	LBR						*R*					
Discretization level →	Case I		Case II		Case III		Case I		Case II		Case III	
Partitioning →	OP	SWP	OP	SWP	OP	SWP	OP	SWP	OP	SWP	OP	SWP
No. of processors ↓												
2	100	98	100	92	100	96	0.003	0.005	0.001	0.003	0.001	0.002
4	100	74	100	77	100	76	0.006	0.009	0.003	0.005	0.002	0.003
8	100	50	100	51	100	50	0.013	0.023	0.007	0.014	0.005	0.009
16	100	34	100	32	100	32	0.021	0.055	0.012	0.033	0.008	0.023
32	100	26	100	23	100	23	0.034	0.119	0.019	0.072	0.013	0.050

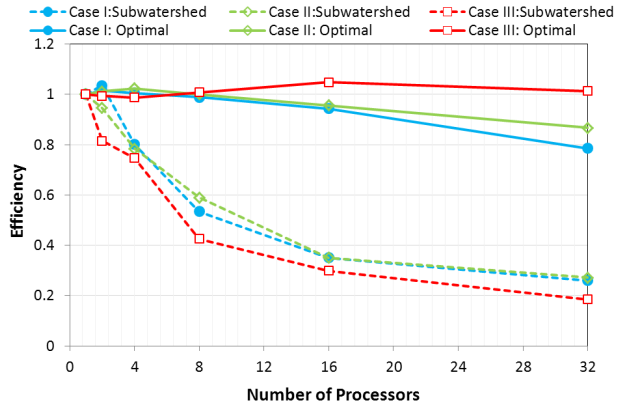

Figure 7: Comparison of parallel efficiencies for optimal (load balance + minimum communication) and sub-watershed based suboptimal partitions for three discretization levels (Table 2). Sub-watershed based parallelization performs poorly with respect to simulation on optimal partitions.

efficiency using the optimal partitioning strategy is close to 1 for up until 32 processors, whereas the parallel efficiency for sub-watershed based partitions reduces to around 0.2. The reduced parallel efficiency is a result of load imbalance and large communication across sub-watershed boundaries relative to optimal partitions. It is to be noted that the goal of this experiment was not to establish that all possible sub-watershed partitions are sub-optimal, but to only highlight that sub-watershed based partitioning strategies are often plagued by load imbalance and large interprocessor communication and are hence likely to underperform in relation to optimal partitions.

Summary and Conclusions

In this study, we presented the efficiency and scalability of the parallel PIHM code for a range of partition configurations and discretizations of the Little Juniata Watershed model domain. The goal was to evaluate both the isolated and integrated role of load balance and minimization of interprocessor communication on the efficiency of parallel hydrologic model simulations. Results suggest that for the same problem size, efficiency of the parallel model decreases with increase in number of processors. More importantly, the rate of decrease in efficiency with increase in the number of processors is not as large for larger problem sizes. This indicates that parallel PIHM simulations based on optimal domain partitions can be expected to exhibit even higher speedups for larger number of processors. Also, parallel efficiency of the model was observed to have a direct correspondence with theoretical metrics of load balance ratio and communication to computation ratio. This implies that these two theoretical metrics can be used to evaluate, screen and identify the best partitions, which can then be later used to perform computationally intensive parallel simulations. Intercomparison of efficiency results between numerous optimal and sub-optimal configurations suggest that for the same problem size, sub-optimal partitions that do not consider either load balance or minimization of interprocessor communication always under-perform relative to optimal ones. More importantly, the decrease in parallel efficiency was much more severe with the same per unit increase in load-imbalance than with increase in interprocessor communication. This suggests that during domain partitioning, load balance should be given priority over minimization of inter-processor communication. Nevertheless, the effect of interprocessor communication on efficiency cannot be neglected. Results also suggest that since not all optimal partitioning algorithms are equally efficient, appropriate partitioning algorithm should be chosen to maximize efficiency. Among the five optimal partitioning algorithms (RGB, RSB, RGB-KL, ML-KL and RSB-KL) considered here, the RSB-KL or ML-KL algorithms were found to outperform others at all four discretization resolutions. For larger partition sizes, ML-KL generally yielded the best partition. As the optimal partition algorithms are determined by the weights of the dual graphs, the paper also demonstrated that hydrologic factors such as the number of hydrologic processes being solved at a discretization element, flow topology, and level of spatial coupling between processes, can be easily incorporated in optimal partitioning algorithms by modulating vertex and edge weights to account for changes in computation load and interprocessor communication respectively. While the gain in parallel efficiency due to the incorporation of hydrologic factors during domain partitioning remains unquantified, the presented methodology is generic and can be applied to both unstructured or structured mesh based distributed models to evaluate the improvements in parallel efficiency. It is to be noted that the results presented in this paper were obtained based on a two months long simulation period. The length of simulation period and the choice of finest discretization resolution were ceiled by the time it took to solve the largest problem on a single processor. Even through the simulation period consisted of numerous precipitation events and intermittent dry periods, one may expect the efficiency results to slightly vary from year to year depending on the precipitation regime and the ensuing hydrodynamics.

Acknowledgement

This study was supported by grants from the National Science Foundation - EAR 0725019 for Shale Hills-Susquehanna Critical Zone Observatory and EAR 1331846 for Calhoun Critical Zone Observatory.

References

1. VanderKwaak JE, Loague K (2001) Hydrologic-response simulations for the R-5 catchment with a comprehensive physics-based model. Water Resour Res 37: 999-1013.

2. Yeh GT, Huang G, Cheng HP, Zhang F, Lin HC, et al. (2006) A first-principle,

physics-based watershed model: WASH123D, 211-244, CRC Press, Boca Raton, FL.

3. Panday S, Huyakorn PS (2004) A fully coupled physically-based spatially-distributed model for evaluating surface/subsurface flow, Adv. Water Resour 27: 361-382.
4. Qu, Y, Duffy CJ (2007) A semidiscrete finite volume formulation for multiprocess watershed simulation, Water Resou Res, 43: W08419.
5. Kumar M, Duffy CJ, Salvage KM (2009) A second-order accurate, finite volume-based, integrated hydrologic modeling (FIHM) framework for simulation of surface and subsurface flow, Vadose Zone J 8: 873-890.
6. Kollet SJ, Maxwell RM (2006) Integrated surface-groundwater flow modeling: A free-surface overland flow boundary condition in a parallel groundwater flow model, Advances in Water Resources 29:945-958.
7. Ivanov VY, Vivoni ER, Bras RL, Entekhabi D (2004) Catchment hydrologic response with a fully-distributed triangulated irregular network model. Water Resour Res 40: W11102.
8. Moore GE (1965) Cramming more components onto integrated circuits. Electronics 38: 114-117.
9. Morton DJ, Zhang Z, Hinzman LD, O'Connor S (1998) The parallelization of a physically based, spatially distributed hydrologic code for arctic regions. Proceedings of the 1998 ACM symposium on Applied Computing, Atlanta, Georgia, United States.
10. Apostolopoulos TK, Georgakakos KP (1997) Parallel computation for streamflow prediction with distributed hydrologic models. J Hydrol 197: 1-24.
11. Cui Z, Vieux BE, Neeman H, Moreda F (2005) Parallelisation of Distributed Hydrologic Model. Int J of Computer Applications in Technology 22: 42-52.
12. Cheng JRC, Hunter RM, Cheng HP, Lin HCJ, Richards DR (2005) Parallelization of the WASH123D code—Phase II: coupled two-dimensional overland and three-dimensional subsurface flows. Impacts of Global Climate Change 1-12.
13. Vivoni ER, Mascaro G, Mniszewski S, Fasel P, Springer EP, et al. (2011) Real-world hydrologic assessment of a fully-distributed hydrological model in a parallel computing environment. J Hydrol 409:483-496.
14. Karypis G, Kumar V (1999) A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J Scientific Comput 20: 359–392.
15. Ashby SF, Falgout RD (1996) A parallel multigrid preconditioned conjugate gradient algorithm for groundwater flow simulations. Nucl Sci Eng 124: 145-59.
16. Kollet SJ, Maxwell RM, Woodward CS, Smith S, Vanderborght J, et al. (2010). Proof of concept of regional scale hydrologic simulations at hydrologic resolution utilizing massively parallel computer resources. Water Resour Res 46: W04201.
17. Kumar M (2009) Toward a hydrologic modeling system, PhD thesis, Pennsylvania State University.
18. Chen X, Kumar M, McGlynn BL (2015) Variations in streamflow response to large hurricane-season storms in a southeastern US watershed. Journal of Hydrometeorology 16: (1).
19. Shi Y, Davis KJ, Duffy CJ, Yu X (2013) Development of a coupled land surface hydrologic model and evaluation at a critical zone observatory. J Hydrometeor 14: 1401-1420.
20. Wang R, Kumar M, Marks D (2013) Anomalous trend in soil evaporation in a semi-arid, snow-dominated watershed. Advances in Water Resources 57: 32-40.
21. Yu X, Lamacová A, Duffy C, Krám P, Hruška J, et al. (2014) Modeling long term water yield effects of forest management in a Norway spruce forest. Hydrological Sciences Journal 60: 174-91.
22. Pacheco PS (2011) An introduction to parallel programming, Morgan Kaufmann Publishers Inc., USA.
23. Hu Y, Blake R (1999) Load balancing for unstructured mesh applications. Parallel and Distributed Computing Practices 2:3.
24. Garey, MR, Johnson DS (1979) Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Co. New York, NY, USA.
25. Kernighan BW, Lin S (1970) An efficient heuristic procedure for partitioning graphs. Bell Systems Tech. J 49: 291-308.
26. Mansour N (1992) Allocating data on the multicomputer nodes by physical optimization algorithms for loosely synchronous computations, Concurrency: Practice and Experience 4: 557-574.
27. Bui TN, Moon BR (1996) Genetic algorithm and graph partitioning. Computers, IEEE Transactions on 45: 841-855.
28. Farhat C (1998) A simple and efficient automatic FEM domain decomposer. Computer and Structures 28: 579-602.
29. Hendrickson B, Leland R (1994) An empirical study of static load balancing algorithms, Scalable High Perf Comput Conf IEEE , 682-685.
30. Williams RD (1991) Performance of dynamic load balancing algorithms for unstructured mesh calculations. Concurrency: Practice and Experience 3: 457-481.
31. Pothen A, Simon DH, Liou KP (1990) Partitioning Sparse Matrices with Eigenvectors of Graphs. SIAM J Matrix Anal and Appl 11: 430-452.
32. Simon HD (1991) Partitioning of unstructured problems for parallel processing. Comput Syst Eng 2: 135-148.
33. Barnard ST, Simon HD (1993) A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. Proceedings of the 6th SIAM conference on parallel processing for scientific computing 711-718.
34. Henderson B, Leland R (1995) A multilevel algorithm for partitioning graphs. Proceeding of Supercomputing ACM, New York, USA.
35. Fowler RF, Greenough C (1998) RALPAR: RAL mesh partitioning program version 2.0, RAL Technical reports, RAL-TR-98-025.
36. Hsieh SH (1993) Parallel processing for nonlinear dynamics simulations of structures including rotating bladed-disk assemblies, Cornell University, Ithaca, NewYork.
37. Hendrickson B, Leland R (1994) The Chaco User's Guide, Version 2.0, Tech Report SAND94--2692 Sandia national laboratories, Albuquerque, NM, USA.
38. Walshaw C, Cross M (2007) JOSTLE: Parallel Multilevel Graph-Partitioning Software - An Overview. In: Magoules F, Mesh Partitioning Techniques and Domain Decomposition Techniques. Civil-Comp Ltd. 27-58.
39. Cohen SD, Hindmarsh AC (1996) CVODE, a stiff/non-stiff ODE solver in C. Computers in Physics 10: 138-143.
40. Gropp W, Lusk E, Skjellum A (1999) Using MPI: Portable Parallel Programming with the Message-Passing Interface. (2nd edn), MIT Press, Cambridge, MA, USA.
41. Brown PN, Byrne GD, Hindmarsh AC (1989) VODE: A variable- coefficient ODE solver. SIAM J Sci Stat Comput 10: 1038-1051.
42. Amdahl G (1967) Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities. AFIPS spring joint computer conference 30: 483-485.
43. Bhatt G, Kumar M, Duffy CJ (2014) A tightly coupled GIS and distributed hydrologic modeling framework. Environ Modell Softw 62: 70-84.
44. Kumar M, Bhatt G, Duffy CJ (2009) An efficient domain decomposition framework for accurate representation of geodata in distributed hydrologic models. International Journal of GIS 23: 1569-1596.
45. Shewchuk JR (1996) Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. Applied Computational Geometry: Towards Geometric Engineering 1148: 203-222.

Appendix

$\vec{F}_0$ Lateral surface flux exchange (LT^{-1})

$\|\vec{F}_1\|$ Lateral surface flux exchange between overland flow and channel (LT^{-1})

$\vec{F}_2$ Lateral groundwater flux exchange (LT^{-1})

$\|\vec{F}_3\|$ Lateral flux exchange between channel and ground water (LT^{-1})

$\|\vec{F}_4\|$ Lateral groundwater flux exchange between sub-channel and triangular watershed element (LT^{-1})

$\vec{F}_5$ Flux exchange between river reaches (LT^{-1})

$\vec{G}_0$ Infiltration/Exfiltration rate (LT^{-1})

$\vec{G}_1$ Recharge flux between unsaturated zone and ground water (LT^{-1})

$\vec{G}_2$ Vertical flux exchange between channel bed and ground water (LT^{-1})

$\vec{G}_3$ Net precipitation flux to the canopy/ground/river (LT^{-1})

$\vec{G}_4$ Evaporation from canopy (LT^{-1})

$\vec{G}_5$ Throughfall drainage (LT^{-1})

$\vec{G}_6$ Snow melt (LT^{-1})

$\vec{G}_7$ Evaporation from overland flow (LT^{-1})

$\vec{G}_8$ Evaporation from upper soil layer (LT^{-1})

$\vec{G}_9$ Transpiration (LT^{-1})

S_1 Sink flux from ground water (LT^{-1})

Author Affiliations

Top

[1]*Nicholas School of the Environment, Duke University, Durham, NC 27708-0328, USA*

[2]*Department of Civil and Environmental Engineering, The Pennsylvania State University, University Park, PA16802, USA*