Domain Partitioning for Implementation of Large Scale Integrated Hydrologic Models on Parallel Processors

Mukesh Kumar & Chris J. Duffy

Abstract

Distributed integrated hydrologic models are data and computationally intensive. In order to perform a high temporal and spatial resolution run of a large scale hydrologic model in feasible time, parallelized versions of hydrologic model can be run on a cluster of parallel processors. An efficient implementation of such a parallelized hydrologic model requires proper partitioning of the model domain. This paper discusses and highlights several hydrologic, architectural and algorithmic issues which need to be incorporated in an efficient domain partitioning for parallel implementation of integrated distributed hydrologic model. Here we also compare a suite of partitioning algorithms, both geometry and graph theory based, in terms of their efficiency in a) minimizing interprocessor communication b) load balancing c) adaptability to constraints and e) capturing actual communication volume. Hybrid algorithms are found to be most effective in minimizing communication volume. But the performance of the algorithms gets adversely affected while trying to satisfy multiple architectural constraints. The algorithms have been implemented on unstructured decomposed domain of Great Salt Lake basin and are discussed vis-à-vis finite volume based Parallelized – Pennstate Integrated Hydrologic Model (P-PIHM).

I. Introduction

Hydrologic models simulate hydrologic state variables in space and time while using information regarding heterogeneity in climate, land use, topography and hydrogeology [Feeze and Harlan, 1969]. These models have inherent advantages over conventional lumped models due to incorporation of natural heterogeneities [Entekhabi and Eagleson, 1989; Pitman et al., 1990] leading to a more physically based simulation of hydrologic processes. Application of these models has varied from synoptic to basin to watershed to hill slope scales. Of late, basin scale models have received significant emphasis by hydrologic modeling community because the land surface units at these regional scales are natural spatial integrators/accumulators of water and associated material transports (Lahmer, 1998). Various physically-based distributed watershed models (e.g. Beven and Kirkby, 1979; Arnold et al., 1991; Liang et. al., 2004; Vivoni et. al., 2004; Ivanov et. al., 2004; Qu, 2004 and Kumar et. al., 2007a) developed over the years simulate major hydrological processes on multiple spatial domains over varied temporal scales with interactions among them spanning from uncoupled to strongly coupled. However, procedures for fully integrating heterogeneities and processes in modeling process for a large scale application is a challenge in itself in terms of having to strike a balance between descriptive detail and computational load. One of the first and important steps in satisfying computational accuracy and efficiency constraint of the model simulation is to perform proper discretization of the model domain into numerous physical subdomains based on hillslopes (Band, 1989), a contour (Moore et al., 1988) or structured/

unstructured grids through a process called domain decomposition and henceforth assigning heterogeneous hydrologic parameters homogenously over the unit cells. Different mathematical formulation of the process equations ranging from finite difference (MODHMS, Panday and Huyakorn, 2004), finite element (FEMWATER, Lin et. al., 1997) to finite volume (PIHM) have been explored synergistically with smart domain decomposition strategies (Kumar et. al., 2007b) to do computationally efficient simulations of coupled nonlinear hydrologic process. But the size of computation increases with increasing spatio-temporal resolution, number of physical processes incorporated in the model and the mathematical complexity of the physical equations and their approximations. This poses considerable challenge to the application of any distributed hydrologic model at higher spatio-temporal resolution. Just to give an idea about the scale of computation, Johnson (2000) took 20 hours of computing time to simulate 20 hours of event time using a two-dimensional numerical model CASC2D (Julien and Saghafian, 1991), a physics based diffusive wave model to simulate the rainfall runoff processes, in Buffalo Creek watershed at a resolution of 72 by 93 meters. Beeson et. al., 2003 in a series of hydrologic simulation experiments over Whitewater River basin in southeastern Kansas using MODHMS model observed that with increase in number of physical processes and complexities there is an exponential increase in simulation time as shown in Fig. 1. Another important aspect in integrated hydrologic modeling which is also computationally intensive is estimation of hydrologic parameters by calibrating the model to observed watershed behavior e.g. streamflow. It is almost computationally impractical to optimize all hydrologic parameter values that define the characteristic of each unit decomposed cell. While comparing various automated calibration techniques over a relatively small catchment at Shale Hills in north Central Pennsylvania using a distributed hydrologic model for a very coarse discretization resolution, Tang et. al., 2006 observed that even the most efficient calibration algorithm may take several days or longer for calibrating an integrated hydrologic model even on state of the art workstation with a 3 GHz processor and 2 Gb of RAM.

This implies that meeting the challenges of integrated hydrologic modeling requires a significant increase in computing power (O'Neill and Steenman-Clark, 2002) which is achieved not only by the faster hardware, but also more importantly by improving the efficiency of faster codes. With the advent of parallel processing architectures, high computing performance can be achieved by parallelization of existing serial integrated-hydrologic-model code. This translates to running different parts of the same model simulation on a network of large number of processors thereby reducing the time needed to obtain solution.

Developing a parallel code requires considerable understanding of hardware architecture, model data structure and interprocessor communications in addition to parallel numerical algorithms to obtain high performance. The primary step in parallelizing a hydrologic model is to map out the problem on multi-processor environment. The problem must be broken down into a set of sub-problems that can be solved concurrently. This strategy of decomposing the modeling problem can be two types viz. task parallelism and data parallelism. In task parallelism a program can be split into independent pieces, often subroutines, which can be assigned to different processors and run concurrently. This essentially means different physical processes viz. surface water model, ground water model, land – atmosphere energy and water exchange,

contaminant and sediment transport model, of an integrated hydrologic model will be solved on different processors. Task parallelism is also called "coarse grain" parallelism because the computational work is spread into just a few subtasks. It is often easier to implement and has less overhead than data parallelism. However since the various physical processes are strongly coupled to each other, the subtasks would need quick interaction between each other which would essentially destroy its “coarse grain” parallel structure. This kind of parallelism will be more suitable for integrated hydrologic models in which large physical components can be considered independent or very weakly coupled. Also since the computation time of various physical processes like overland flow and groundwater flow are significantly distinct owing to their varied time scales, time of computation on various processors will be different. The performance of the code is then limited by the slowest processor output. The remaining idle processors do no useful work. Task parallelism also limits the number of processors that can be utilized thus reducing the scalability of parallelization. In other parallelization strategy of data parallelism the same code segment runs concurrently on each processor, but each processor is assigned its own part of the data to work on. In this case the decomposed modeling domain on which the physical constitutive relationships are defined is allocated in chunks to different processors. Notably, data parallelism also provides single flow of control defined by Single-Program-Multiple-Data Model where the code is identical on all processors. Parallelized hydrologic model code obtained by following either of the aforesaid discussed strategies can be implemented on parallel shared memory processors or distributed memory processors. In shared memory computers, all processors have access to a single pool of centralized memory with a uniform address space. Any processor can address any memory location at the same speed so there is Uniform Memory Access time (UMA). Processors communicate with each other through the shared memory. Codes are also easier to program on it however they don’t scale much and architecture is limited to only a handful of processors. Contrary to this, though programming a parallel code on a distributed memory is complicated but these are quite scalable and can have support of even thousand processors. The total memory is partitioned into memory that is private to each processor and so the communication between processors takes Non-Uniform Memory Access time (NUMA). This means that farther the communicating processor, longer is the access time.

So an integrated hydrologic model code that is parallelized based on data-parallelism scheme on a distributed memory can be expected to be scalable. However the speedup obtained from the parallel code will strongly depend on how the mapping of the model domain is performed on different processors.

This paper studies and compares the domain partitioning algorithms vis-à-vis Parallelized PennState Integrated Hydrologic Model (P-PIHM). Section II introduces the basic concepts of P-PIHM. Section II discusses some limited applications of domain partitioning in hydrology and will identify the factors that need to be addressed by a good domain partitioning algorithm incorporated in a parallelized integrated hydrologic model. Section III discusses several existing domain partitioning algorithm and suggests ways to incorporate the factors which will determine the efficiency of respective partitioning algorithms for hydrologic applications. Both the strengths and weaknesses of these algorithms will be discussed. Section IV discusses the results of application of partitioning algorithms in Great Salt Lake River Basin. Section V discusses the

limitations of the existing algorithms and will make draw conclusions from the experiments presented in this paper.

II.1. Parallelized Pennstate Integrated Hydrologic Model

The parallelized version of PIHM (Qu and Duffy, 2007; Kumar et. al., 2007a) called P-PIHM (Kumar et. al., 2007c), solves coupled physical hydrologic processes distributed over TINs. The basic idea of P-PIHM and PIHM is to first identify the physical hydrologic relationships which can be represented in form of partial or ordinary differential equations (PDE or ODE). By applying divergence theorem over a control volume, governing PDEs can be transformed to semi-discrete ODEs while ensuring mass conservation. The model is designed to capture "dynamics" in multiple processes while maintaining the conservation of mass at all cells, as guaranteed by the finite volume formulation. The advantage of finite volume formulation is that the user can incorporate the desired number of processes simply by setting on/off switches prior to simulation. In addition to this, it ensures mass conservation and also has ability to handle discontinuous solutions (Leveque, 2002). The "control-volume" in the finite volume formulation is a prismatic or linear physical element which is also called model kernel with all the constitutive relationships identified. Fig 2 shows a typical kernel defined on a triangular and a linear element (corresponding to rivers only) along with the interacting processes. The conservation laws that are conveniently derived from the physical relationships approximate the average state over the kernel volume [Leveque, 2002].

The relevant ODEs defined on a kernel (Qu, 2004) are shown below in table 1. In Table 1, Q_s^{ij} is surface flow from element i to its neighbor j. P_o, I and E_o are precipitation, infiltration and evaporation respectively. Q_{oc} describes interaction between overland flow and channel routing. q^0 is internal flux between unsaturated zone and saturated zone. I and ET_s are incoming infiltration and outgoing evapotranspiration at land surface, respectively. E_c is evaporation from channel. Q_g^{ij} is lateral groundwater flow from element i to its neighbor j. Q_l is vertical leakage through an underlying confining bed. Q_{gc} is discharge/recharge from/to aquifer to/from channel. Q_{in} and Q_{out} are flow in and out of a channel segment. Δw is snow melting rate, which is also an input to overland flow.

Table 1: ODEs for the hydrologic processes defined on a kernel

Process	Governing equation/model	Original governing equations	Semi-discrete form ODEs
Channel Routing	St. Venant Equation	$\frac{\partial h}{\partial t}+\frac{\partial (uh)}{\partial x}=q$	$\left(\frac{d\varsigma}{dt}=P_c-\sum Q_{gc}+\sum Q_{oc}+Q_{in}-Q_{out}-E_c\right)_i$
Overland Flow	St. Venant Equation	$\frac{\partial h}{\partial t}+\frac{\partial (uh)}{\partial x}+\frac{\partial (vh)}{\partial y}=q$	$\left(\frac{\partial h}{\partial t}=P_o-I-E_o-Q_{oc}+\sum_{j=1}^{3}Q_s^{ij}\right)_i$

Unsaturated Flow	Richard Equation	$C(\psi)\frac{\partial \psi}{\partial t} = \nabla \cdot (K(\psi)\nabla(\psi + Z)$	$\left(\frac{d\xi}{dt} = I - q^0 - ET_s\right)_i$
Groundwater Flow	Richard Equation	$C(\psi)\frac{\partial \psi}{\partial t} = \nabla \cdot (K(\psi)\nabla(\psi + Z)$	$\left(\frac{d\zeta}{dt} = q^0 + \sum_{j=1}^{3} Q_g{}^{ij} - Q_l + Q_{gc}\right)_i$
Interception	Bucket Model	$\frac{dS_I}{dt} = P - E_I - P_o$	$\left(\frac{dS_I}{dt} = P - E_I - P_o\right)_i$
Snow melt	ISNOBAL	$\frac{dS_{snow}}{dt} = P - E_{snow} - \Delta w$	$\left(\frac{dS_{snow}}{dt} = P - E_{snow} - \Delta w\right)_i$
Evapotran-spiration	Pennman-Monteith Method	$ET_0 = \frac{\Delta(R_n - G) + \rho_a C_p \frac{(e_s - e_a)}{r_a}}{\Delta + \gamma(1 + \frac{r_s}{r_a})}$	$\left(ET_0 = \frac{\Delta(R_n - G) + \rho_a C_p \frac{(e_s - e_a)}{r_a}}{\Delta + \gamma(1 + \frac{r_s}{r_a})}\right)_i$

Note that set of all the above differential equations defined on a kernel can be represented as

$$My' = f(t, y; x(y)) \qquad (1)$$

where M is the identity matrix, y is the state variable column vector with each row corresponding to ξ (unsaturated mean moisture depth), ζ (saturated mean moisture depth), h (overland flow depth) and ς (channel flow depth) respectively. t is time and x is the forcing i.e. P_o, I, E_o, Q_l, ET_s and E_c.

Due to inherently different time scales present in different processes processes, the resulting system is likely to be "stiff". An ODE solver called CVODE [Cohen and Hindmarsh, 1994] from Suite of Nonlinear and Differential/Algebraic equation Solvers [SUNDIALS, 1994], developed at the Lawrence Livermore National Laboratory (LLNL, 2005) is used to solve large scale "stiff" ODE system.

As is evident from the process equations listed in Table 1, the magnitude of the state variables on a kernel is dependent on those in the neighboring kernels and must be updated at each calculating time interval. On a serial computer this data transfer is accomplished by writing to and reading from memory. However when we map this computational grid to a parallel computer, two vertices joined by an edge and not owned by the same processor must communicate to exchange values. If, as is typically the case, communication is more expensive than computation. So a domain mapping strategy that minimizes it is desirable. Of course we could assign the entire grid to a single processor and have no communication at all, but that wouldn't be an effective use of the parallel machine since one processor would do all the work while the others remained idle. We must therefore also observe the important constraint that each processor should be assigned about the same amount of work and therefore (in the simplest case) the same number of vertices. This means a classic domain-decomposition strategy as discussed in Gilbert et. al., 1995 or Simon, 1991 fails to provide performance gains.

In P-PIHM each processor does a portion of the work that is assigned to it while interacting with its peers to exchange data. Implementation of the parallelized model is performed using parallel version of CVODE solver also called PVODE. PVODE uses MPI and a revised version of the vector module in CVODE to achieve parallelism and portability. PVODE is intended for SPMD (Single Program Multiple Data) environment with distributed memory in which all vectors are identically distributed across processors. However mapping of parts of the model domain should be devised for proper load balancing and efficient communication between jobs on different processors

II.2. Domain Partitioning

In one of the preliminary investigation on domain partitioning for parallelization of tRIBS (Ivanov et. al., 2004), Vivoni et. a. (2005) integrated hydrologic model, chunks of tasks on subbasins were used to be fed to different processors for calculation in parallel. The problem with this is highlighted in Fig. 3 which shows how the number of unit cells or TINs generated within each sub-basin is significantly different. This essentially translates to different computation time on different processors. The efficiency of parallelization will ultimately be defined by the processor with largest computational load and is slowest. Cui. et. al., 2005 also partitioned the watershed into subbasins but tackled the problem of load imbalance by redistribution of load between processors using sending by pairs, sending circularly and sending by percentage methodology to send data from overloaded to underloaded processors in order to balance load among processors. The experiment is interesting however it involves lot of communication between processors which could have been completely avoided by partitioning the domain such that load is balanced. More so these strategies consider hydrologic independence between sub-basins which will be true only when the distributed hydrologic model doesn't takes care of groundwater flow and flow through river from on sub-basin to another. If these hydrologic interactions are taken into account, it will be very difficult to keep track of the load as well as associated communication from and to various unit cells of the sub-basin when they are being shared all around the processors in order to balance load. So the aforesaid partitioning is quite ineffective not only because the size of sub-basins can be significantly different leading to load imbalance between various processors but also the communication cost can be large. Also such strategies are adhoc and provide solution to only a given problem at hand. They will have to be reposed in order to take into account the heterogeneity of communication between various unit cells in the basin and architectural heterogeneity.

Partitioning should reflect the basic requirements of parallel processing like

a) To ensure that all processors have the same amount of work to do or to perform Load Balancing. Load balancing is the technique of evenly dividing the workload among the processors. Load balancing is important because the total time for the program to complete is the time spent by the longest executing thread. This means that a perfectly load balanced code will have least computation time given a particular number of processors.

b) To minimize interprocessor communication. Communication time is the time it takes for processes to send and receive messages. The cost of communication depends on the amount and frequency with which data is communicated between processors and latency

and bandwidth of the interconnection network. Latency is the time it takes to set and prepare a complete communication for a message length of size zero, where bandwidth is the actual speed of transmission, or bits per unit time. This time must be minimized to get the best performance improvements from a parallel program. Even though the number of communication is lot less than the computations, since the cost of accessing memory on other processors is about 10 to 1000 times larger than that of accessing it locally, minimizing communication becomes crucial. By overlapping communication and computation, idle time of the processors can be minimized. This involves computations performed way inside the solution domain which is temporarily independent of the changes in state variable taking place at the interfaces while the peripheral neighbors take part in communication.

Factors affecting load balance and interprocessor communication

a) ***Hydrologic Factors***

i) Number of hydrologic processes incorporated in the model, whose value at a particular discretized element also depends on its values in its surrounding; determine the amount of interprocessor communication.

ii) Process Coupling: It is increasingly common to couple multiple physical phenomena e.g sediment, contaminant and water transport into a single simulation. The amount of communication is strongly dependent on the method of coupling incorporated in the model. It is of two types: a) artificial coupling, where processes operate at time steps consistent with their own appropriate temporal scales and are updated with other interacting processes as and when there time steps coincide. This strategy is followed in tRRIBS (Ivanov et. al., 2004) and MIKE SHE hydrologic model. *Prima facie* it appears that because of the limited amount of sharing between faster and slower processes, communication requirements will be smaller in artificial coupling. However, as has been noted in Panday and Huyakorn, 2004 and Kumar et. al., 2007a, artificial coupling strategy can be more computationally taxing because of its difficulty in convergence (Perkins and Koussis, 1996; Beven, 1985 and Refsgaard and Storm, 1996) which can only be offset by taking very smaller simulation steps. The other coupling strategy is b) natural coupling, where the simulation proceeds at self adaptive time steps depending on the characteristic time scales of the interacting system. This strategy is followed in P-PIHM hydrologic model. Particularly in situations where a faster process like overland flow is not happening for most of the simulation period because of any of the numerous reasons like less precipitation forcing, high infiltration rate and dried water table, natural coupling will be all the more efficient as it will adaptively revert to larger time steps in those situations as has also been witnessed by Kumar et. al., 2007a..

iii) Topology: Number of neighbors of a particular element is determined by the shape of unit elements and the topological relations between different feature types. The maximum number of communication interface for a grid will be equal to 4* (Number of state variable whose value depends on the states in neighboring cells) + River upstream-downstream topology dimension. For triangular unstructured grids, number of communicating interface between elements will be 3.

iv) Heterogeneous Computational Load: Processors can take significantly different time to solve system of ODEs defined on two different model kernels using a same ODE solver depending on the characteristic time scale and degree of stiffness of the ODE system in different regions of the model domain which in turn depends on the values of forcing, parameters and the physical processes acting on each model kernel. For example, a smaller Manning's coefficient and a large infiltration rate will further decrease the time scale of overland flow process in a model grid with respect to ground water flow leading to more number of iterations in the solutions of coupled ODEs corresponding to both the processes. Infact with increasing stiffness of the coupled system, the number of iterations for convergence increases (Kumar et. al., 2007a.). This means that in a distributed model, the heterogeneity in forcing and parameters will keep influencing the synchronicity at each solution time step across different processors. We note that the computational load will be varying spatio-temporally.

v) Heterogeneous Communication: Amount of communication between neighboring model kernels located on different processors is determined by the number of processes defined on the model kernel which need information about state variables from the adjacent kernel. Table 2 shows that the amount of communication performed between neighboring elements in a simplified two layered (unsaturated and groundwater) conceptualization of P-PIHM, at each simulation time step is different in different parts of the model domain. Similar will be the case with other hydrologic models. Fig 4 represents the unit model element and the communicating processes between them for two hydrologic models – one based on grids and other on TINs. This heterogeneity in communication needs to be incorporated while partitioning the model domain. Also, the difference in timescale of various hydrologic processes like overland flow and groundwater flow can be used to our advantage in order to further improve the efficiency of the code by performing updates of groundwater variable on the boundary cells at relatively longer time intervals with respect to the model time interval. For example, if the overland flow model simulation is being carried out at time step *n*, groundwater flow across the subshed boundaries can be calculated at 5**n* only. The underlying assumption here would be that a change in the subsurface storage is very slow relative to the overland flow. However, this methodology will be more fitting to models based on artificial coupling only whose limitation have already being pointed out in the discussion above.

Table 2: Size of communication packet for different elements of the model domain

Elements	Shared Processes	Total Communication
Triangular Elements besides Subshed boundary	Sub-surface flow	1
Triangular Elements besides River	Upstream flow, Downstream flow, Subsurface flow between triangular elements on either side of river, Leakage/Base flow from/to the river to triangular element, Overland flow to/from river	5

Triangular elements	Overland flow, Sub-surface flow	2

b) ***Architectural factors***

i) Interconnect property and type: The interconnection network are wires and cables though which the multiple processors of a parallel computer are connected to each other and to the memory units. The communication time is dependent upon the specific type of the interconnection network and its properties like latency, bandwidth, diameter and degree. Latency is the delay on a network that occurs while a data packet is being stored and forwarded. Bandwidth determines the amount of data that can be sent through a network connection. Diameter is the distance between two processors that are farthest apart. Degree determines the number of communicating wires coming out of each processor. A smaller latency and diameter, and a larger bandwidth and degree are desired for shortest communication time.

Topology of the interconnection network also determines the chances of network congestion when a message is sent between distant processors. This is because while the interconnection is transferring messages, the wires are rendered unavailable to transmit other messages. Commonly used network topologies are Bus, Cross-bar switch and Hypercube. Bus based interconnections are more prone to have contention for access than cross-bar switch. The advantage with Hypercube interconnections is larger degree with increasing size of network. Hence when network congestion is important, weighting messages by the number of wires they use will lead to better domain mappings to distributed processors.

Many at times, the distributed processor are also connected heterogeneously, with communication occurring within a group and between groups of processors. The disparity between communication time between the local and remote connections needs to be incorporated in a domain partitioning strategy. Heterogeneity in data transfer can also be because of different network interfaces and protocols.

ii) Heterogeneous Processors Speed: Heterogeneous clusters can have individual nodes with varying processor speeds. This is particularly likely for Beowulf cluster of PCs built with commodity-off-the-shelf equipment where faster machines with larger memories are continually added to the system or replaced for slower nodes. The consequence of this is to be able to delay the obsolescence of older technologies thus further reducing the cost of high performance computing. In order to minimize the idle processor time for computing on heterogeneous clusters, the work over them should be so distributed such that no processor is waiting for the completion of another. Thus the partitioning algorithm for a heterogeneous processor configuration should be able to incorporate asymmetric load balancing.

Considering the architectural and hydrologic factors that influence load balance and communication between processors, the problem now is that of how to decompose the mesh into subdomains while incorporating the needs of an efficient parallel computation. This essentially translates to a set of minimization (or maximization) problem as discussed above, given an arbitrary number of balancing constraints like heterogeneous communication and processor speeds. Many of these partitioning

problems can be formulated in terms of an undirected communication graph. The communication graph describes the relationship of computation on the mesh by connecting unit elements which share information between each other. If the numerical algorithm (finite difference, element or finite volume) has a node based data structure, meaning that the state variables e.g. hydraulic heads in a hydrologic model are defined on the mesh nodes and fluxes along the edges, then any updates of state variable over time will also require data from neighboring nodes. Therefore the communication graph in this case is essentially the computational mesh itself, with mesh nodes being the graph vertices and edges of the mesh being the edges of the graph. The other kind of data structure can be element based where the state variables are defined on the elements and fluxes are calculated across the interfaces of neighboring elements. In this case vertices of the communication graph are essentially the centroid of the elements, and the edge of the graph is the connecting segment joining two vertices lying in the neighboring elements that share a face with each other. Such a graph is called the *dual graph* of the mesh. This approach is explained in detail in Hu and Blake, 1999. Fig 5 shows the dual graph for unit elements for three different hydrologic models viz. ModHMS (Panday and Huyakorn, 2004), PIHM and ELCIRC (Zhang, et. al., 2004) respectively on an experimental rectangular model domain. We note that unit element shapes for each of these models is different viz. rectangular (structured mesh) for ModHMS, triangular (unstructured mesh) for PIHM and mixed mesh for ELCIRC.

The problem of efficient portioning can now be defined on the dual graphs. Given a dual graph G with n weighted vertices and m weighted edges, the objective is to divide the vertices into p partition sets in such a way that the sum of the vertex weights in each set is as close as possible and the sum of the weights of edges crossing between sets is minimized. The weights on the vertices and edges are generally proportional to the computation load on the elements and communication amount across the element face respectively. The posed problem is NP-complete and so it's hard to obtain the global optimum solutions. Therefore several near-optimal approximate, probabilistic and heuristic techniques have been explored to solve the problem (Walshaw and Cross, 1999; Hu and Blake, 1999; deCougny et. al., 1994; Simon, 1991).

III. Domain Partitioning Algorithms

Some of the prominent heuristic methods and their characteristics are briefly discussed below. Many of these are bisection based which essentially means dividing the domain into two subdomains and to perform divisions recursively on the obtained subdomains.

a) Inertial Bisection: The recursive inertial bisection (RIB) algorithm (Hendrickson and Liland, 1994) is a coordinate based method which tries to find a principal axis hyperplane of the communication graph thus dividing it into two parts. The principal axis is the line from which the sum of the squares of distances of the mesh nodes is smallest. The method is rotationally invariant unlike other geometric bisection algorithms like recursive coordinate bisection algorithm (Williams, 1991). The algorithm has a low complexity of $O(n)$.

b) Greedy Method: This algorithm (Farhat, 1998) is one of the simplest and fastest graph based partitioning method. Assuming that desired number of partitions is *p* and the total number of nodes is *n*, first $\frac{n}{p}$ nodes are coded in a partition *i* by including all the neighbors of a node location with minimum number of neighbors and also the neighbor's neighbors. The process is repeated for rest of the domain until all the nodes have been assigned to a partition. The algorithm has a low complexity of $O(n)$.

c) Graph Bisection: The *recursive graph bisection* (RGB) algorithm (Williams, 1991) first finds a set of pseudo peripheral nodes (PPNs) which are basically the two vertices that are the furthest apart (their distance is called the diameter of the graph). Then, starting from either of the PPNs , half of the graph nodes that are closer to either of the PPNs are assigned to two separate partitions. This process is then recursively executed on each of the subdomains. The graph bisection algorithm has a complexity of $O(n)$.

d) Spectral Bisection: The *recursive spectral bisection* (RSB) algorithm (Pothen *et. al.*, 1990; Simon, 1991) is a discrete optimization method. By assigning each nodes of the graph with a value of either 1 or -1, and defining the edge-cut for the bisection by

$$|E_c| = \frac{1}{4} \sum_{i \leftrightarrow j,(i,j) \in V} (x_i - x_j)^2 \qquad (2.a)$$

where $i \leftrightarrow j$ is an edge connecting the nodes i and j respectively in partition *V*, the communication can be minimized by minimizing E_c while ensuring

$$\sum_{i=1}^{n} x_i = 0 \qquad (2.b)$$

Also noting that all the nodes take the value of 1 or -1, the sum of the squares should be n, the number of nodes. This gives the extra constraint

$$\sum_{i=1}^{n} x_i^{\,2} = n \qquad (2.c)$$

Since

$$\sum (x_i - x_j)^2 = \sum (x_i^{\,2} + x_j^{\,2}) - \sum 2 x_i x_j = x^T D x - x^T A x$$

where D and A are diagonal matrix, with degree of the nodes on the diagonal, and adjacency matrix respectively. Defining *Laplacian matrix* of the graph as *L= D – A*, the aforesaid optimization problem with constraints can be rephrased as

$$|E_c| = \frac{1}{4} x^T L x \qquad (3)$$

The eigenvector corresponding to second lowest eigenvalue of the matrix L, also called Fiedler vector, is used to divide the nodes into two halves. The procedure can then be repeated on each of the subdomains.

e) Multilevel partitioning: The spectral bisection method discussed above is very computationally intensive because of the eigenvector solution. Multilevel methods (Barnard and Simon, 1993) speeds up the computation of Fiedler vector still generating high quality partitions. The algorithm is based on the multilevel approach normally and consists of three phases viz.

i) coarsening phase: the original graph is reduced into a levels of successively coarser graphs

ii) partitioning phase: the coarsest graph is partitioned into p parts

iii) uncoarsening and refinement phase: the partitioning of the coarsest graph is interpolated to a finer level graph and refined. The process is repeated till refinement reaches the original graph level.

The coarsening is achieved by choosing the maximal independent set (MIS) as the vertices of the coarse graph. MIS is a set of vertices such that no two of them are connected by an edge and if the addition of even a single vertex will violate this criteria. The edges of the coarse graph are weighted to reflect the number of edges in the original graph. By using several levels of coarsening a much smaller graph can be obtained which can be easily and rapidly partitioned by other graph partitioning methods like graph or spectral bisection. Infact Karypis and Kumar, 1995 observed that the choice of partitioning algorithm applied at coarsest scale has almost no bearing on the final quality of partition because of refinement performed in uncoarsening phase. In uncoarsening and refinement phase, the partitioning information are transferred up through the levels to the original graph using methods like eigenvector interpolation.

f) K – L Algorithm: The K-L (Kernighnan-Lin) algorithm (Kernighan and Lin, 1970) is an iterative algorithm that tries to iteratively improve random load balanced partitions. The algorithm tries to exchange the vertices from one partition of the graph to the other in order to reduce the edge cut till all the nodes of the smaller partition have been swapped. The procedure is repeated even if there are no more improvements made and continues even when the highest gain may be negative thus enhancing its ability to climb out of local minima. The algorithm has a complexity of $O(|E|)$ where E is the number of edges. For large graphs, the algorithm is quite inefficient and so is now used for local refinement of partitions obtained by algorithms discussed above.

g) Hybrid Algorithms: These are basically combination of two or more algorithms, one suited for global partitioning and the other for local partitioning only due to its computational intensiveness, that work in unison to give better results. K-L algorithm have often been used as the local search algorithm to improve partitions generated by ML algorithm (Karypis and Kumar, 1995), RSB and RGB algorithm (Fowler and Greenough, 1998).

e) Other partitioning algorithms: Many other geometric and graph partitioning methods exist like coordinate bisection method (Williams, 1991), linear method, simulated

annealing (Mansour, 1992), generic algorithms (Bui and Moon, 1996) etc. While coordinate bisection method is anisotropic, simulated annealing and genetic algorithm based methods are computationally intensive though they also produce high quality partitions. Algorithms which take into account the physical equations for finite element based solution of PDEs (deCougny et. al, 1994;. Vanderstraeten and Keunings, 1995)have also been developed

IV. Results

All the above algorithms for domain partitioning are applied on unstructured domain decomposition of Great Salt Lake Basin. The mesh has been generated by using topographic and hydrologic features as internal boundary constraints. The topographic features such as subshed boundary essentially divides the basin in four subsheds viz. Weber River, Bear River, Utah Lake and Western Desert with corresponding areas varying from 6413 km^2 to 49117 km^2. The discretized domain is then partitioned in order to assign different computational model kernels to different processors. As is evident from the huge variation in area of the subshed, if the average resolution of unit elements within each subshed is same then the time of computation for each subshed will be vastly different. We note that the measure of amount of communication used in the following discussion is the cumulative weighted edge cut at each interface unless otherwise mentioned.

a) ***Homogeneous Communication****:* Fig 6 shows the partitioning of the decomposed domain using inertial, greedy, recursive graph bisection, recursive spectral bisection, random Kernighan-Lin method and Multi-Level method based on spectral algorithm. As can be seen from the Fig 6, only RSB and ML generate all contiguous partition which is paramount particularly in fully coupled hydrologic modeling where the granularity is very fine. Non-contiguous partitions also suffer from message congestion as they will have to generally travel longer distances and larger number of neighbors to fetch data. Disconnected and long partitions with large surface to volume ratio, as is the case with partition generated using inertial bisection algorithm is not desirable. However, it also has a relatively smaller number of neighboring partitions which remarkably decreases the message startup time. Partitions produced by RGB are found to be compact, rough and disconnected. Domains generated by both RSB and ML algorithms are smooth, compact and connected. Quantitative comparison of the methods is shown in the Fig 7. Among the basic methods, KL algorithm outperforms other algorithms in minimizing interface communication whereas ML method based on spectral bisection algorithm outperforms the rest in minimizing congestion and message startup time which essentially depends on the average number of neighboring partitions. Ranking wise, RSB algorithm is found to perform best on an average for both criteria. This is because the RSB algorithm captures the global property of the dual graph by calculating Fiedler vectors. The hybrid variant of RGB, RSB and ML algorithms with KL based refinement leads to improved performance in terms of decreased communication and number of neighboring partitions, except for ML_KL method where number of neighboring partition increases w.r.t ML method. Depending on the relative time taken by the parallel hardware architecture in communication and startup, the number of times synchronization is forced between

processors and new communication is initiated (will also depend on the software coding strategy), and the length of the model simulation, decisions can be made by the user to give preference to a particular property and hence a particular algorithm. RSB_KL outperforms all the considered algorithms in minimizing communication volume. We note that KL refinement on RGB, RSB and ML increases the number of neighboring partitions. For larger domain size (number of graph nodes) ML algorithm is found to be computationally efficient than RSB based methods (Karypis and Kumar, 1995). So computation time saved due to less communication in RSB_KL algorithm can be offset by time it takes to derive the partition in the first place for very large graphs. This will particularly be crucial while performing spatio-temporal adaptive refinement/de-refinement of decomposed domain and during dynamic partitioning of the model domain due to spatio-temporal heterogeneity in computational load, as in these cases partitioning code has to be called numerous times.

b) ***Heterogeneous Communication:*** In real hydrologic applications, the communication requirements across processors will be generally heterogeneous. As shown in Fig 8, the amount of communication between neighboring elements in different parts of the model domain is different. This heterogeneity can be incorporated in graph partitioning algorithms by assigning weights to the edges of dual graph proportional to the amount of communication. A detailed discussion of RSB and ML algorithms where vertex and edge weights are modified to account for computational and communication heterogeneity is discussed in Hendrickson and Leland, 1995a,b respectively. Fig 9(a) shows the mapping of decomposed GSLB into 16 partitions using hybrid RSB_KL and ML_KL algorithms while considering heterogeneity in communication. The obtained partition has far less communication with respect to the case with no weights assigned to graph edges. Infact Fig 9(a) clearly shows the tendency of partition boundaries to align along the subshed boundaries because of relatively lower communication requirement across them. In order to study the effectiveness of the algorithm at various scales, the domain is decomposed into 979, 1295, 2232 and 4566 triangles respectively. Accounting for heterogeneity consistently leads to decrease in communication volume to the order of 75 to 85 %. Also RSB_KL algorithm performs better than ML_KL at almost all scales. Weight assignments performed in Table 1 according to the process interaction that are shown in Fig 8 can be further made favorable by decreasing weights for groundwater flow particularly along the subshed boundaries, which are by-the-way often considered as groundwater divide also, to take advantage of the fact that groundwater processes have relatively longer time scales at most of the places and can be considered to be non-dependent on the groundwater head of the neighboring element that lies across the watershed-divide. However, we note that such partitioning will introduce error in modeling and should be only implemented after studying the tradeoff between computational accuracy and load.

c) ***Heterogeneous Processors:*** If the processors used for computation have different speeds, then the load divided between them can be balanced by distributing the number of elements in proportion to processors speed. Fig 10 shows the partition of GSLB into 16 partitions using RSB algorithm on homogeneous and heterogeneous processors. The

size of the partitions that are assigned to a faster processor increases in proportion to the processor speed.

d) ***Message Congestion in Interconnect:*** Interconnection topology and sequence of assignment of partitions to processors significantly affect message contention costs. Since network contention renders the network unavailable to transmit any more messages, it should be minimized. Hendrickson et. al., 1996 addressed this issue for ML and RSB algorithm using a method called terminal propagation which basically improves the data locality by including the processor location information in partitioning. The goal is to partition the domain such that processors sharing information are mapped closer together in the interconnect topology. This results in a message traveling between two processors to traverse least distance. Fig 11 shows the advantage of termination propagation in reducing the hop cost for spectral partitioning algorithm. However, this also results in increase in the number of edge costs and hence the communication volume. So depending on the relative time spent in starting send/receive operation and the time the message takes to traverse between processors , terminal propagation should be taken into account.

e) ***Refinement of Global partitioning algorithms:*** As shown in Fig. 7, global partitioning methods are observed to perform better in unison with local refinement methods such as using KL algorithm. The refinement can be carried out after each time a graph is recursively bisected or on the final partition. Also, refinement can be performed either on the boundary nodes only or can involve repeated passes over all vertices to find a better configuration. Fig 12 shows the relative communication (characterized by edge cuts and hypercube hops), computation (characterized by number of internal vertices) and start-up message time (characterized by average number of neighboring sets) for RSB algorithm by refining the partitions by KL algorithm with different number of refinement sweeps. Edge cuts determines the communication volume while Hypercube hops based communication measure also takes into account the architectural distance between processors which a particular message will have to traverse. With the application of KL algorithm on the boundary vertices only, there is a significant improvement in communication and startup time criteria. The performance further improves marginally when internal vertices are also considered in refinement. However, larger number of sweeps on internal vertices slows the process.

f) ***Limitation of communication measure:*** Effectiveness of any of the partitioning algorithms can only be translated in real modeling application if the communication measure takes into account all the hydrologic and architectural factors discussed above. But first and foremost the basic assumption of the communication measure which have been implemented in most partitioning softwares i.e. use of weighted edge-cut should be true. Hendrickson and Kolda, 2000 pointed out that existing measure to calculate communication volume can fail in certain situations. For example, edge cuts of the dual graph might correspond to transferring the same data from one partition to the other. This is generally the case with converging or diverging edges connecting nodes which belong to different partitions. The idea is shown in Fig 13 using a schematic two partition domain. Though grids A and B in Fig 13 need to send only one information each to the neighboring partition, the number of edge-cuts accounted for during partition is 3 and 2

respectively. But this also depends on the fact how communications between different paritions are coded in the model. The communication measure to derive partitioning, and the data structure of the buffer that is actually used to perform communication through MPI should be consistent. Also as have been pointed out earlier, multiple objectives that will ultimately determine the performance of partitioning in a real application can not be satisfied all together. An alternate single objective measure can be derived that uses the startup or latency time with communication volume. Fowler and Greenough, 1998 use one such measure called Communication Time which can be defined by

$$\text{Communication Time (CT)} = \sum_{i=1}^{p} N_i t_{start} + 8 t_{send} I_i = N_T t_{start} + 8 t_{send} \sum_{i=1}^{p} I_i \qquad (4)$$

Where

t_{start} is time to initialize communications

t_{send} is time to send one byte

N_i is the number of neighboring subdomains

I_i is the number of interface nodes associated with partition *i*

N_t is the total number of neighbors

This measure assumes that only one message can be active at a time and the total time spent in communication is just the sum of all the communication times for individual partitions (Fowler and Greenough, 1998). The communication time measure based partitioning will obviously be subjective to a given interconnect topology and parallel hardware architecture. Fig 14 shows the relative communication time taken by four basic algorithms discussed in this paper. The typical startup and send time used in the calculation is of Intel parallel supercomputer ipsc/860 such as on in Oakland National Laboratory (data used from Fowler and Greenough, 1998). The processor nodes are assumed to be connected in hypercube architecture. For larger number of partitions, RSB proves to be most efficient. We note that even this measure doesn't takes into account message contention, multihop costs, message length dependent buffering and coding strategy.

The partitions in this paper have been generated using codes from some of the state-of-the-art partitioning softwares like CHACO (http://www.cs.sandia.gov /CRF/chac_p2.html), JOSTLE (http://staffweb.cms.gre.ac.uk/~c.walshaw/jostle/), METIS (http://glaros.dtc.umn.edu/gkhome/views/metis/) and RALPAR (http://www.softeng.cse. clrc.ac.uk/ralpar/).

V. Conclusions

This paper highlights several issues which need to be taken care of while doing efficient partitioning for a parallelized hydrologic model. These issues can be hydrologic in terms of time scale of hydrologic processes which determines the computational load at a model kernel and also the frequency of communication needed, number of interacting processes, coupling behavior, numerical solution strategy and unit element shapes of

decomposed domain. Nine domain partitioning algorithm have been implemented on a unstructured grid domain of Great Salt Lake basin. Recursive spectral bisection algorithm refined by KL algorithm is found to outperform the other algorithms in minimizing communication. The present best-partitioning measures are multi-objective in nature and the tradeoffs have to be accounted for between objectives before any particular partitioning can be applied in real hydrologic model simulation.

References

Arnold, J.G., Williams, J.R., Griggs, R.H., Sammons, N.B., 1991. A basin scale simulation model for soil and water resources management. Texas A&M Press, p. 255.

Barnard, S.T. and Simon, H.D., "A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems", Proceedings of the 6th SIAM conf. on parallel processing for scientific computing", p711-718, 1993.

Band, L. E., Topographic partition of watersheds with digital elevation models, Water Resour. Res., 23(1), 15–24, 1986b.

Beven, K., Kirkby, M.J., 1979. A physically based, variable contributing area model of basin hydrology. Hydrol. Sci. Bull. 24, 43–69.

Beven, K.J., 1985, Distributed models, in M.G. Anderson and T.P. Burt (Eds) Hydrological forecasting, Wiley, Chichester.

Bui, T.N. AND B. R. Moon, Genetic algorithm and graph partitioning, IEEE Transactions on Computers, 45 (1996), pp. 841-855.

Cohen, S.D. and A.C. Hindmarsh, *CVODE User Guide*, Lawrence Livermore National Laboratory technical report UCRL-MA-118618, September 1994.

Cui. Z., B. E. Vieux, H. Neeman, and F. Moreda, Parallelization of Distributed Hydrologic Mode, International Journal of Computer Applications in Technology, 22, 1, 2005

DeCougny, H.L, K.D. Devine, J.E.Falherty, R.M. Loy, C. Ozturan and M.S.Shephard, Load balancing for parallel adaptive solution of partial differential equations, Applied Numerical Mathematics, 1994, pp. 157-182

Entekhabi, D., Eagleson, P.S., 1989. Land surface hydrology parameterization for atmospheric general circulation model including subgrid scale spatial variability. J. Climate 2, 816–831.

Farhat, C., *A simple and efficient automatic FEM domain decomposer*, Computer and Structures, 28 (1988), pp. 579-602

Fowler, R. F. and C. Greenough, RALPAR: RAL mesh partitioning program version 2.0, RAL Technical reports, RAL-TR-98-025, 1998.

Freeze, R.A., Harlan, R.L., 1969. Blueprint for a physically-based, digitally-simulated hydrologic response model. J. Hydrol. 105, 237–258.

Hammond, S., Mapping unstructured grid computations to massively parallel computers, PhD thesis, Rensselaer Polytechnique Institute, Dept. of Computer Science, Troy, NY, 1992

Hendrickson, B. and T. Kolda, Graph partitioning models for parallel computing, Parallel Computing, Vol. 26, 12, 2000

Hendrickson, B. and R. Leland, An empirical study of static load balancing algorithms, In Proceedings of the Scalable High Performance Computer Conference, pp. 682-685,IEEE, 1994

Hendrickson, B. and R. Leland, An improved spectral graph partitioning algorithm for mapping parallel communications, SIAM J. of Sci. Computation, 16, 1995a

Hendrickson, B. and R. Leland, A multilevel algorithm for partitioning graphs, Proc. ACM/IEEE conference on supercomputing, 1995b.

Hluchy, L., V. D. Tran, J. Astalos, M. Dobrucky, G. T. Nguyen, D. Froehlich: Parallel Flood Modeling Systems. International Conference on Computational Science ICCS'2002, pp. 543-551.

Hu, Y. and R. Blake. *Load balancing for unstructured mesh applications*. Parallel and Distributed Computing Practices, 2(3), 1999.

Ivanov, V.Y., Vivoni, E.R., Bras, R.L. and Entekhabi, D., 2004, The catchment hydrologic response with a fully-distributed triangulated irregular network model. Water Resources Research (In Press).

Johnson, B.E., Julien, P.Y., and Watson, C.C. (2000) "Development of a Storm Event Based Two-Dimensional Upland Erosion Model (CASC2D-SED)," *American Water Resources Association (AWRA)* February 2000

Julien, P. Y., and Saghafian, B. (1991). "CASC2D user's manual - A two dimensional watershed rainfall -runoff model". *Civil Eng. Report, CER90-91PYJ-BS-12*, Colorado State University, Fort Collins, Fort Collins, CO

Karypis, G. and V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, Technical Report TR95-035, Department of Computer Science, University of Minnesota, 1995.

Kernighan, B. W., and S. Lin, An efficient heuristic procedure for partitioning graphs, Bell Systems Tech. J., 49 (1970), pp. 291-308.

Kumar, M., G. Bhatt and C.J. Duffy, 2009a, The Role of Physical, Numerical and Data Coupling in a MesoscaleWatershed Model, Advances in Water Resources, (In Review)

Kumar, M., G. Bhatt and C.J. Duffy, 2009b, An efficient domain decomposition framework for accurate representation of geodata in distributed hydrologic models, International Journal of GIS, v.23

Kumar, M and C. J. Duffy,2009c A large scale implementation of Parallelized Pennstate integrated hydrologic model. In preparation.

Lahmer, W.(1998). Macro- and Mesoscale Hydrological Modelling in the Elbe River Basin. In: Proceedings of the International Conference 'Catchment Hydrological and Biochemical Processes in Changing Environment' in Liblice, Czech Republic, September 22-24, 1998, p.57-61.

Leveque, R.J., 2002. Finite Volume methods for hyperbolic problems. Cambridge University Press.

Liang, X., Guo, J., Leung, L.R.,2004, Assessment of the effects of spatial resolutions on daily water flux simulations, Journal of Hydrology, 298, 287-310

Lin, H. C., D. R. Richards, G. T. Yeh, J. R. Cheng, H. P. Cheng, and N. L. Jones, FEMWATER: A three-dimensional finite element computer model for simulating density-dependent flow and transport in

variably saturated media, Report CHL-97-12, U.S. Army Corps of Engineer, 3909 Halls Ferry Road, Vicksburg, MS 39180-6199, September, 1997

Macks,A.; Heterogeny in a Beowulf, High Performance Computing Systems and Applications, 2002. Proceedings. 16th Annual International Symposium on 16-19 June 2002 Page(s):42

Mansour, N., *Allocating data the multicomputer nodes by physical optimization algorithms for loosely synchronous computations*, Concurrency: Practice and Experience, 4 (1992), pp. 557-574.

Moore I. D. , E. M O'Loughlin, Burch. A., Contour-based topographic model for hydrological and ecological applications, Earth Surface Processes Landforms,13,305-320, 1988.

Namburu, R. R., D. Turner, and K. K. Tamma. An effective data parallel self-starting explicit methodology for computational structural dynamics on the Connection Machine CM-5. International Journal of Numerical Methods in Engineering, 38:3211-3226, 1995.

O'Neill, A. and Steenman-Clark, L. 2002. The computational challenges of Earth-system science. Philosophical Transactions of the Royal Society of London, Series A 360: 1267-1275.

Panday, S., and P.S. Huyakorn (2004). A fully coupled physically-based spatially-distributed model for evaluating surface/subsurface flow, *Advances in Water Resources*, *27*, 361-382.

Perkins, S.P. and A.D. Koussis, 1996, Stream–aquifer interaction model with diffusive wave routing. Journal of Hydraulic Engineering, American Society of Civil Engineers 122 (4), 210–218.

Pitman, A.J., Henderson-Sellers, A., Yang, Z.L., 1990. Sensitivity of regional climates to localised precipitation in global models. Nature 346, 734–737.

Pothen, A., D. H. Simon and K. P. Liou, Partitioning sparse matrices with eigenvectors of graphs, SIAM Journal of Matrix Analysis and Applications, 11 (1990), pp. 430-452.

Qu, Y., 2004, An integrated hydrologic model for multi-process simulation using semi-discrete finite volume approach.", PhD Thesis, 2004, PSU

Rebaine, A., F. Fortin and A. Benmeddour, Parallelization of a Finite Volume CFD Code, ICPPW 04

Refsgaard, J. and B.Storm, 1996, MIKE SHE, pp 809-846 in Computer Models for Watershed Hydrology, Water Resource Public., Fort Collins, Colorado

Simon, H. D., Partitioning of unstructured problems for parallel processing, Computer Systems in Engineering, 2 (1991), pp. 135-148.

Tang, Y., P. Reed and T. Wagener, 2006, How effective and efficient are multiobjective evolutionary algorithms at hydrologic model calibration, HESS, 2, 2465-2520.

Vivoni E.R., V.Y.Ivanov, R.L. Bras, and D, Entekhabi (2004), Generation of triangulated irregular networks based on hydrological similarity, Journal of hydrologic engineering, *9*, 4. 288-302.

Vivoni., E.R., S. Mniszewski, P. Fasel, E.S. Springer, V. Y. Ivanov and R. L. Bras, Parallelization of fully distributed hydrologic model using Sub-Basin Partitioning, Fall AGU Annual Meeting, 2005.

Walshaw, C. and M. Cross. Parallel Mesh Partitioning on Distributed Memory Systems. In B. H. V. Topping, editor, Computational Mechanics Using High Performance Computing, pages 59-78.

Saxe-Coburg Publications, Stirling, 2002. (Invited Chapter, Proc. Parallel & Distributed Computing for Computational Mechanics, Weimar, Germany, 1999).

Williams, R. D., *Performance* of dynamic load balancing algorithms for unstructured mesh calculations, Concurrency: Practice and Experience, 3 (1991), pp. 457-481.

Zhang, Y.-L., Baptista, A.M. and Myers, E.P. (2004) "A cross-scale model for 3D baroclinic circulation in estuary-plume-shelf systems: I. Formulation and skill assessment". Cont. Shelf Res. 24: 2187-2214.

Vanderstraeten, D. and R. Keunings, Optimized partitioning of unstructured finite element meshes, International Journal of Numerical Methods in Engineering, 38: 433-450, 1995

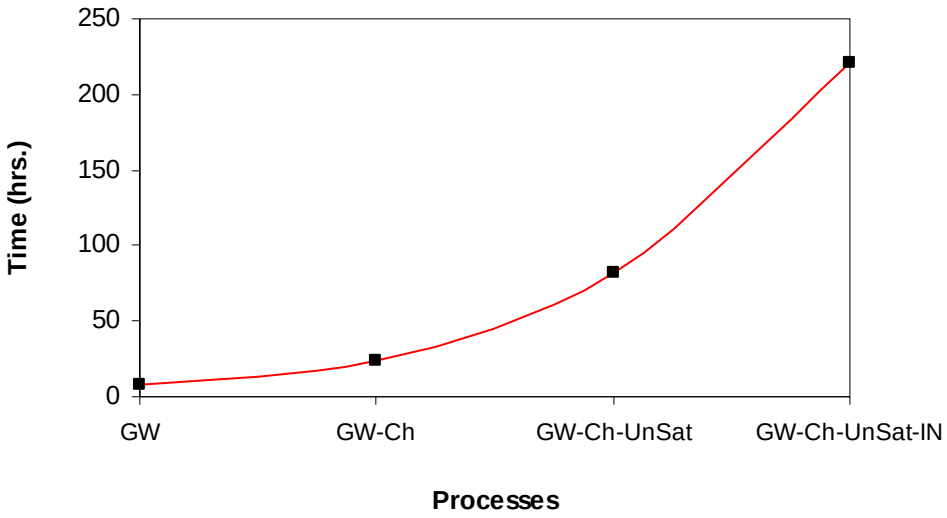

Fig 1: Sharp increase in ModHMS hydrologic model simulation time at White Water Basin, Kansas with increasing model complexity. Abbreviations: GW (Ground Water flow), Ch (Channel flow), Unsat (Unsaturated zone flow), IN (Interception).

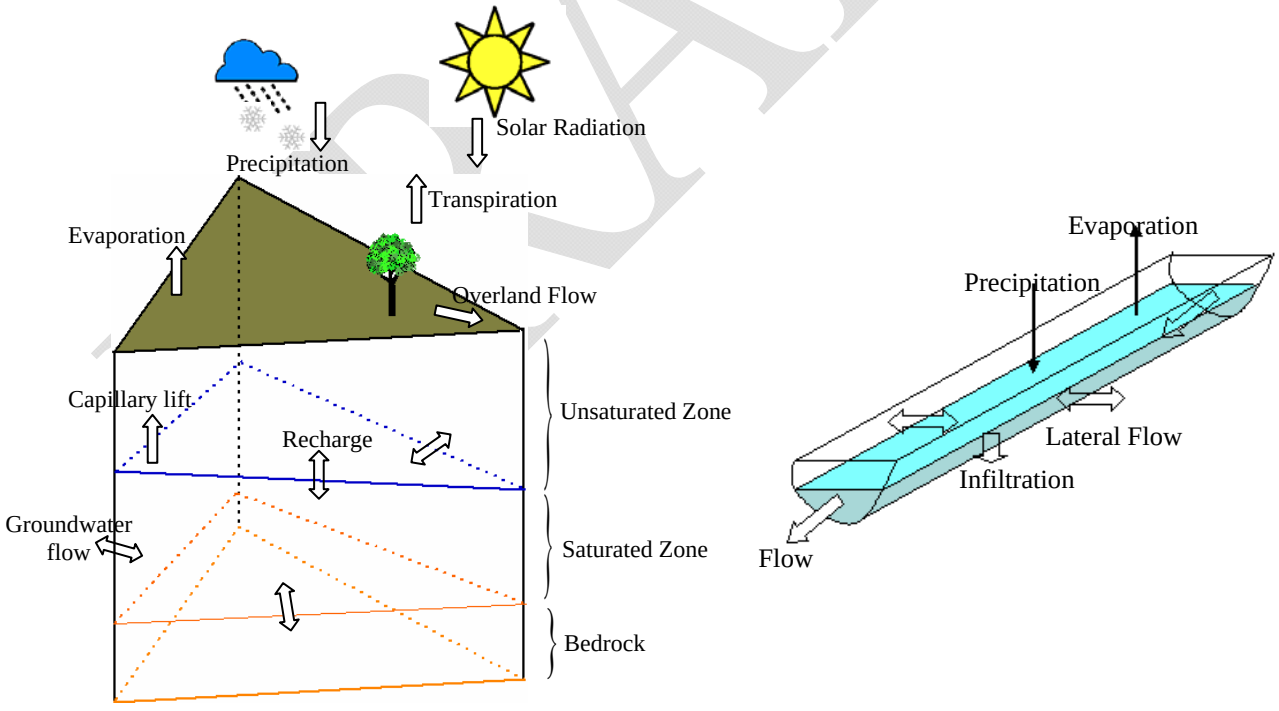

Fig. 2: Interacting hydrologic processes on each prismatic element (left) and on each linear river element (right). ODEs corresponding to all these processes is termed as model "kernel".

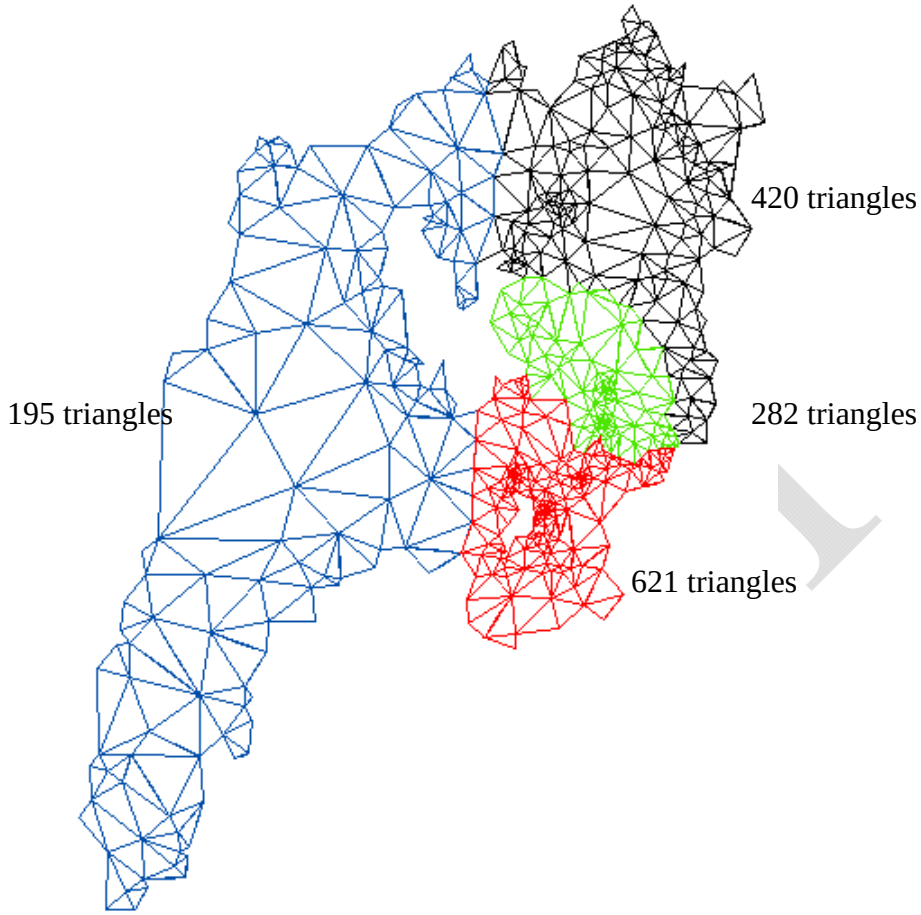

Fig. 3: Partitioning the model domain according to subshed leads to load imbalance. Note that the number of unit model kernels and so the number of ODEs defined on different subsheds is different.

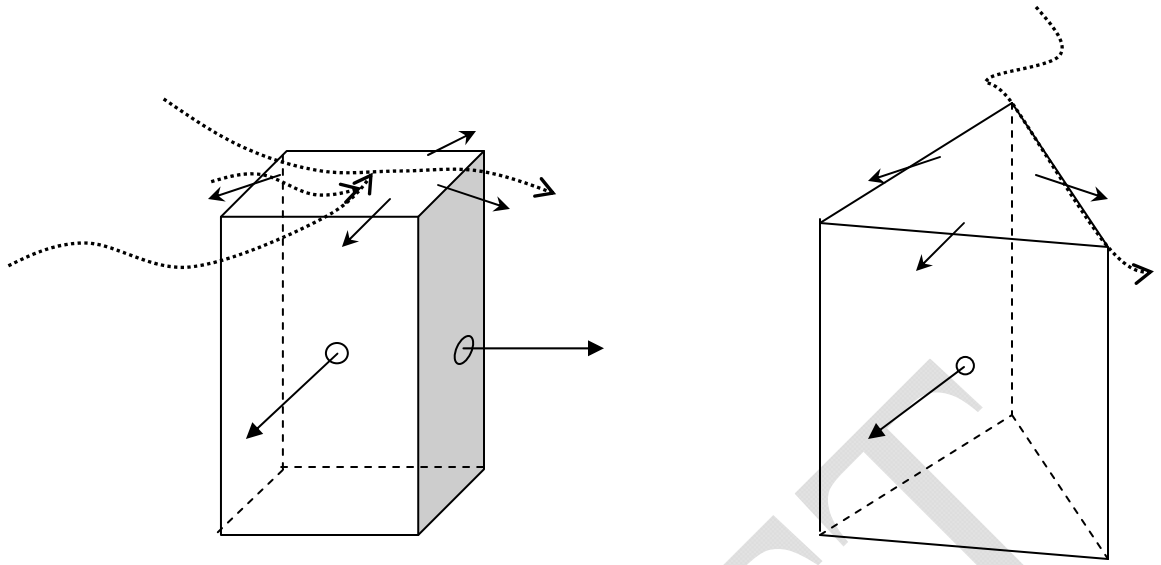

Fig. 4: Model unit kernel for a) ModHMS and b) PIHM. Note that the maximum possible amount of communication across the kernel face will be: (Number of River segments entering and going out through kernel face) + (NLayer for subsurface flow) + (4 units of Overland Flow communication) for ModHMS. For PIHM, maximum communication across as face will be: 2 (Upstream and Downstream) + 2 (Leakage/Baseflow to adjacent subsurface element, weirflow/overland flow between river and adjacent element) + (NLayer for subsurface flow) + (3 units of Overland Flow communication).

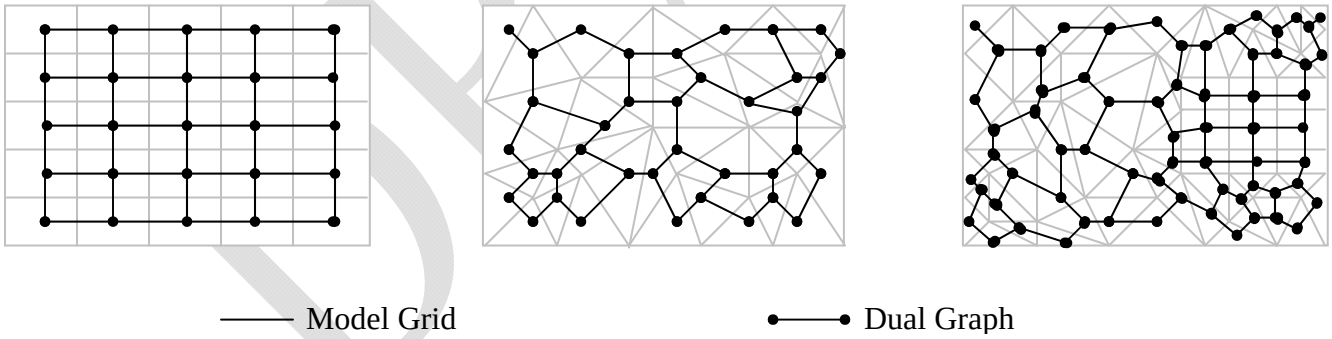

Fig. 5: Dual graphs and the discretized model domain for a) ModHMS [solution strategy = finite difference] b) P-PIHM [solution strategy = finite volume] and c) ELCIRC [solution strategy = finite difference and volume]. Decomposed unit element shape in model (a) = Structured Grid, (b) = Unstructured Grid (Triangles) and (c) = Unstructured Grid (Triangles and Quadrilateral). Note that all of above three models are block centered.

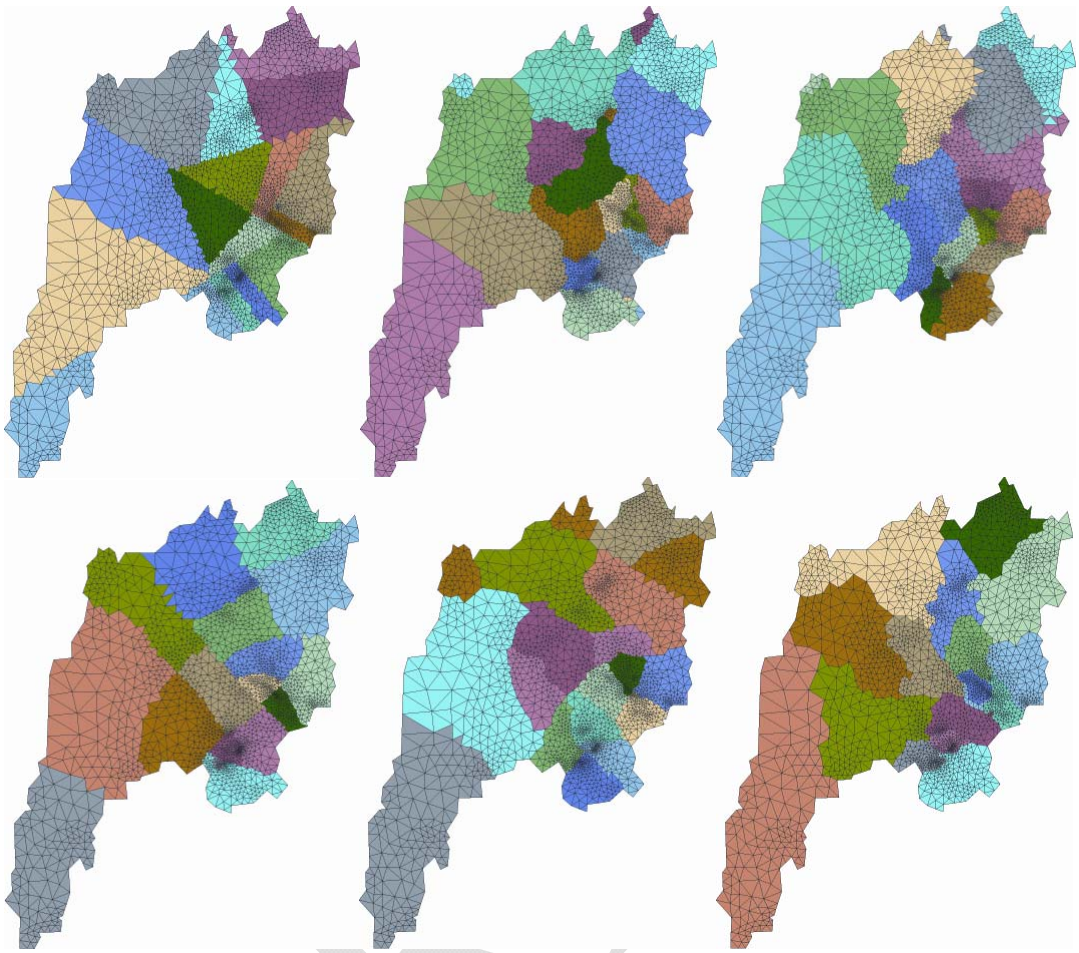

Fig 6: Partitioned domain for Great Salt Lake Basin (total number of unstructured grids = 4566) into 16 partitions. Algorithm used in partitioning is a) Inertial b) Greedy c) Recursive graph d) Recursive spectral e) Random KL and f) Multi-Level. Assumption: Homogeneous communication across the unstructured grid edges.

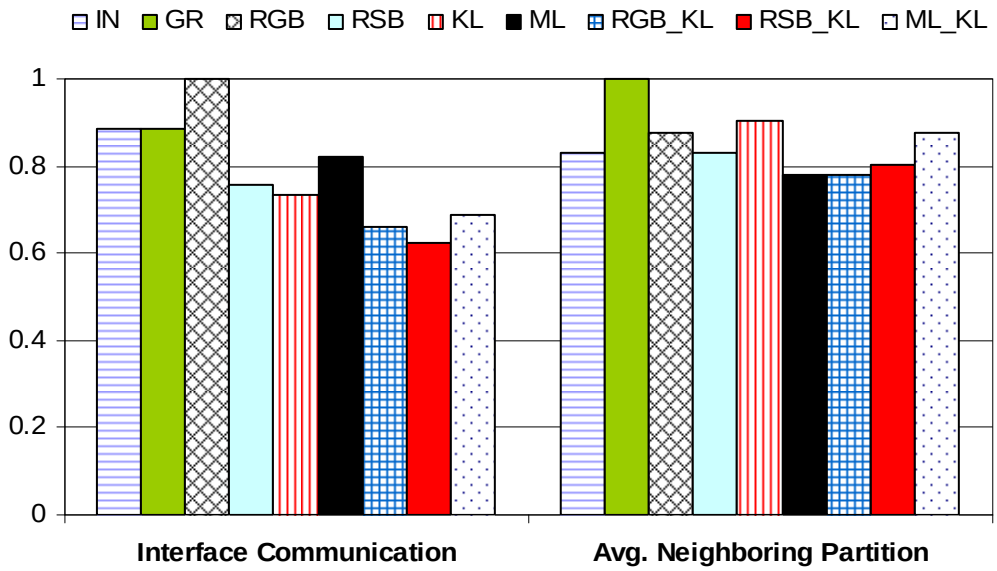

Fig 7: Relative values of communication and average number of neighboring partitions for different partitioning algorithms. IN is Inertial, GR is Greedy, RGB is Recursive Graph Bisection, RSB is Recursive Spectral Bisection, KL is Kernigham-Lin, ML is Multi Level (based on RSB) and RGB_KL, RSB_KL and ML_KL are hybrid methods with location refinement being performed using KL method. Hybrid methods perform best atleast in minimizing communication volume.

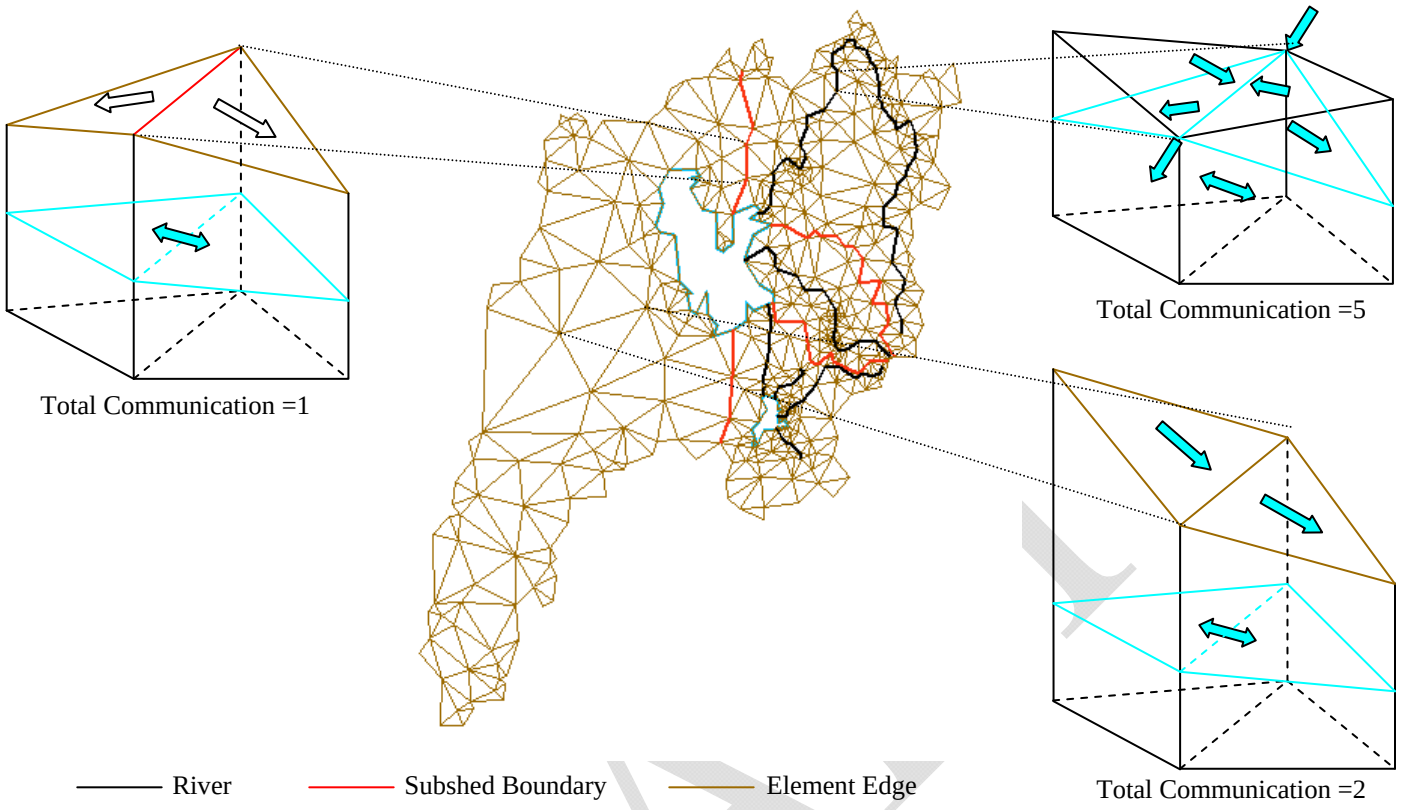

Fig 8: Heterogeneous communication exists in different parts of the model domain because of existence of disparate hydrologic features with different kinds of interacting processes.

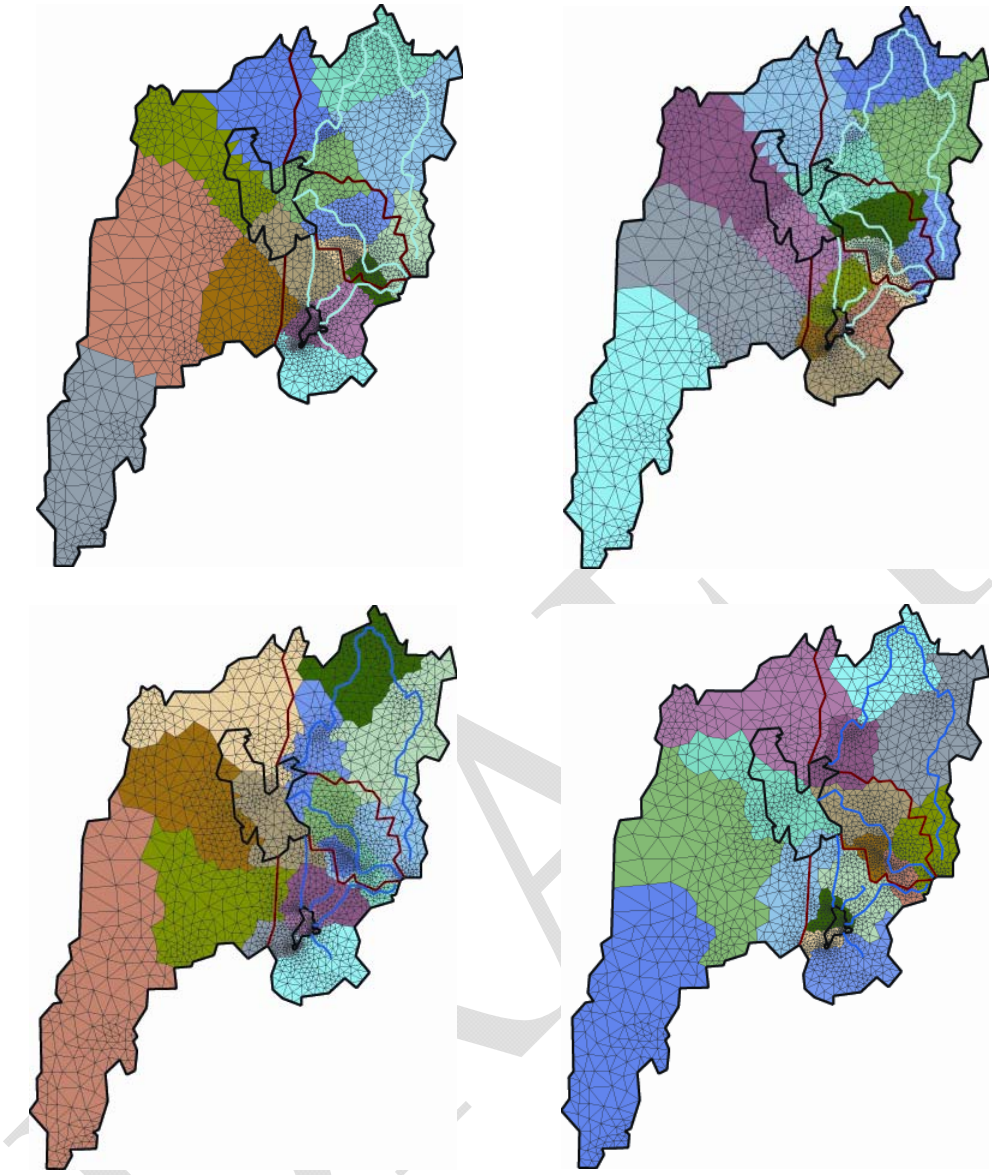

Fig 9 (a): Mapping of GSLB into 16 partitions without (left) and with (right) heterogeneous communication taken into account using RSB_KL (top) and ML_KL (bottom) algorithm respectively. Note the alignment of partition boundary to subshed boundary (particularly in bottom right figure) because of less communication across them.

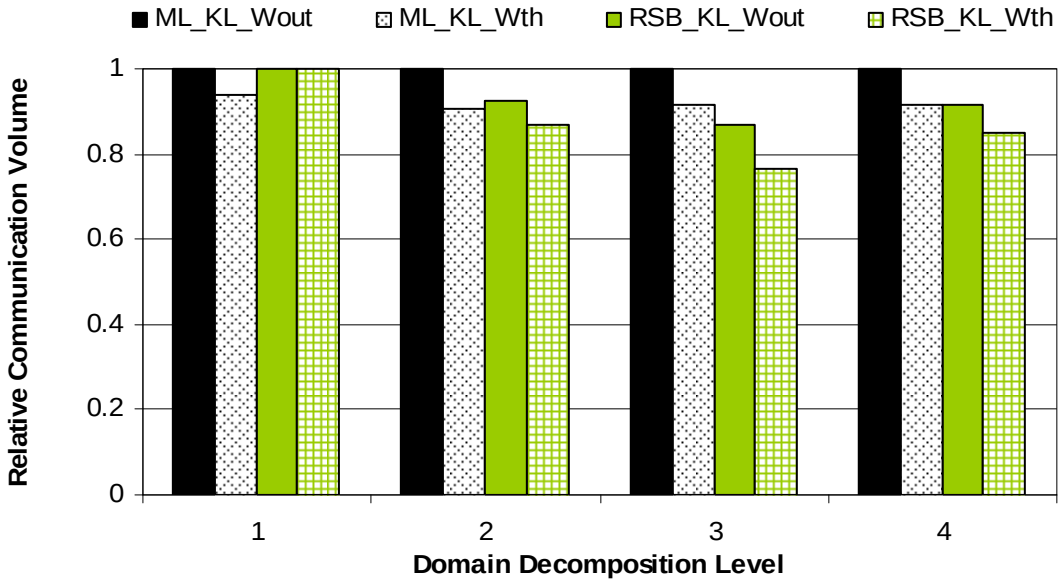

Fig 9(b): Percentage edge cuts for without and with heterogeneous communication taken into account in ML algorithm. Increasing decomposition level 1,2,3 and 4 denote higher level of discretization of the model domain with 979, 1295, 2232 and 4566 unit elements respectively. Wth and Wout stands for "With heterogeneity in communication consideration" and "Without heterogeneity in communication consideration" respectively

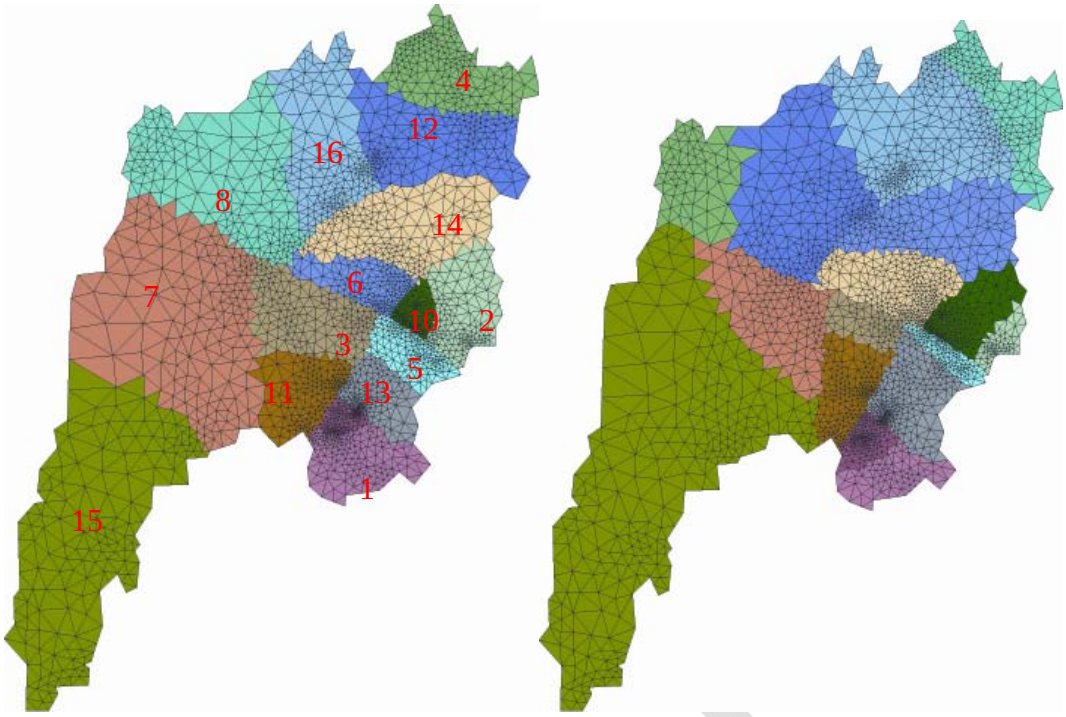

Fig 10: Partitioning of GSLB using RSB algorithm into 16 partitions. In the right figure, heterogeneous processor speeds have been considered with the relative speeds assigned as partNo(1,2,3,4) = procSpeed(1), partNo(5,6,7,8) = procSpeed(2), partNo(9,10,11,12) = procSpeed(3) and partNo(13,14,15,16) = procSpeed(4). Note the increase in size of partitions that are assigned to faster processors.

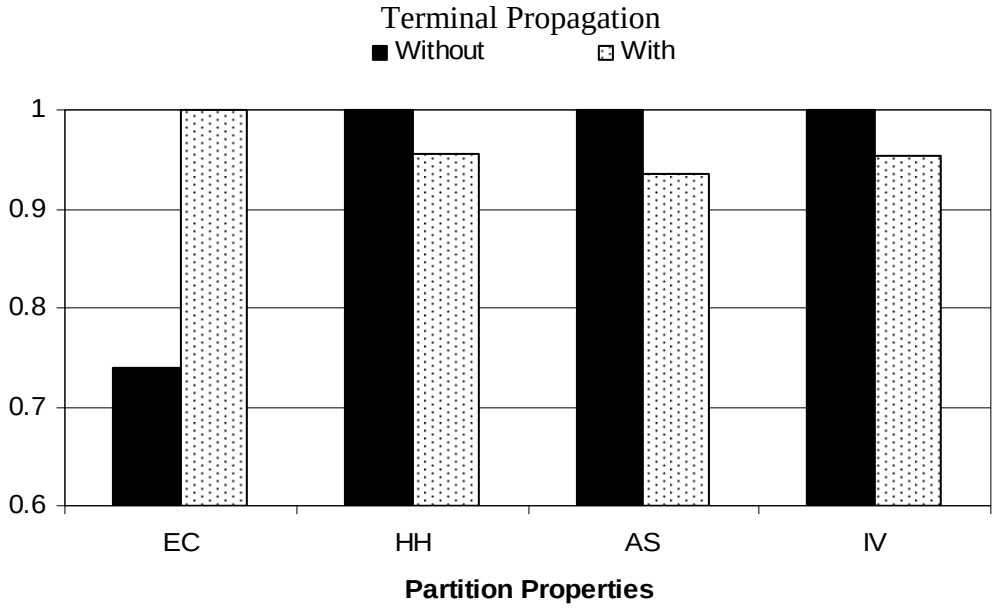

Fig 11: Terminal propagation reduces hypercube-hops. EC = Edge-Cuts, HH = Hypercube-Hops, AS = Average Adjacent Sets and IV = Internal Vertices. One the one hand accounting for terminal propagation in partitioning reduces HH, on the other it increases the EC. Tradeoff have to be evaluated before a real model simulation.

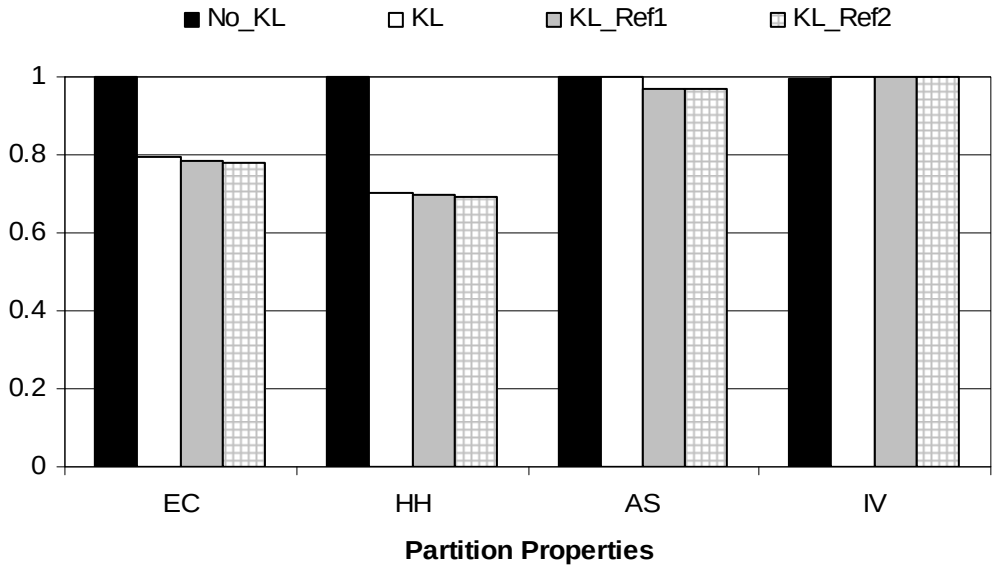

Fig. 12: Relative communication (EC = Edge-Cuts, HH = Hypercube Hops), computation (IV = Internal vertices) and message start-up time (AS = Number of Adjacent Sets) for partitions generated by global RSB algorithm and further refinement by local KL algorithms with refinement performed on boundary vertices (KL) and on all vertices with one and two sweeps (KL_Ref1 and KLRef2) respectively.

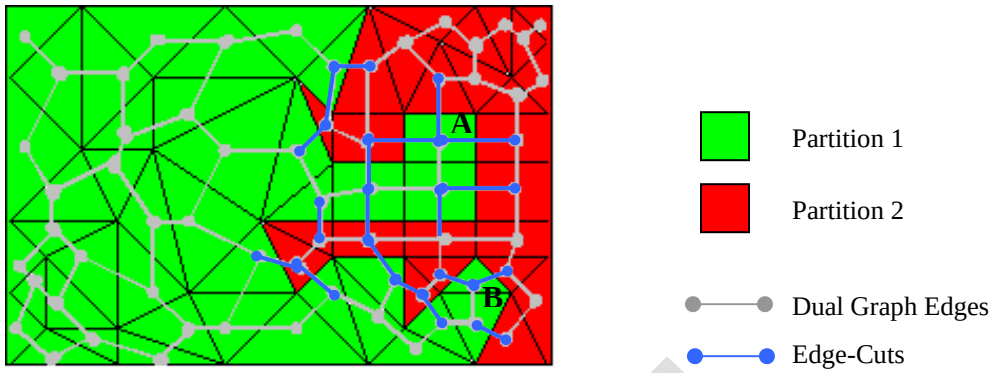

Fig 13: Partitioned domain for mixed unstructured grid (shown in Fig. 5) into two partitions. Note that number of grids in green partition which share a face with red partition is 12, however number of edge cuts is 19.

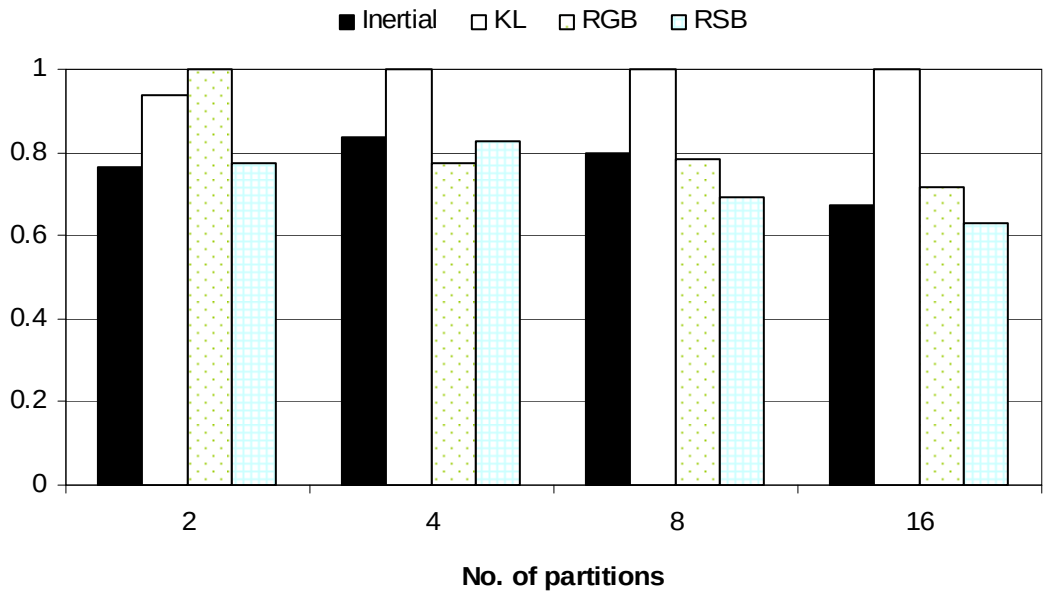

Fig: 14: Comparison of Total Communication time (startup + exchange) for Inertial, KL with random initial partition, RGB and RSB algorithms. Total Communication time presents a single objective measure that can be optimized by partition, but this also doesn't take into account message contention.