

Parallelization of the WASH123D code—Phase I: 2-Dimensional Overland and 3-Dimensional Subsurface Flows

Jing-Ru C. Cheng^{a*}, Hsin-Chi Lin^{b†}, Hwai-Ping Cheng^{b‡}, Robert M. Hunter^a, David R. Richards^c, Gour-Tsyh Yeh^{d§}

^aMajor Shared Resource Center (MSRC)
U.S. Army Engineer Research and Development Center (ERDC)
Vicksburg, MS 39180-6199, USA

^bCoastal and Hydraulics Laboratory
U.S. Army Engineer Research and Development Center
Vicksburg, MS 39180-6199, USA

^cInformation Technology Laboratory
U.S. Army Engineer Research and Development Center
Vicksburg, MS 39180-6199, USA

^dDepartment of Civil and Environmental Engineering
University of Central Florida, 4000 Central Florida Blvd., Orlando, FL 32816-2450, USA

The parallel WASH123D, which is supported by the DoD CHSSI (Department of Defense Common High Performance Computing Software Support Initiative), is designed to solve watershed problems on scalable computing systems. WASH123D is a first-principle, physics-based model to compute water flow and/or contaminant and sediment transport within a watershed system. In the WASH123D model, a watershed is conceptualized as a coupled system of one-dimensional (1-D) channel network, two-dimensional (2-D) overland regime, and three-dimensional (3-D) subsurface media. It aims to address the environmental issues concerning both water quantity and quality. To reach numerical solutions with reasonable and tolerable computer time for simulations that embrace large meshes, numerical algorithm improvement and code parallelization are two essential tasks. Mathematically, 1-D channel flow and 2-D overland flow are described with the St. Venant equations, which are solved with either the Semi-Lagrangian or the Eulerian finite element method. The 3-D subsurface flow is governed by the modified Richards equation,

*This work was supported in part by a grant of computer time from the Department of Defense High Performance Computing Modernization Program at the U.S. Army Engineer Research and Development Center (ERDC), Major Shared Resource Center (MSRC).

†This work was supported in part by DoD CHSSI Project Number 9976.

‡The work was supported in part by the U.S. Army Engineer District, Jacksonville (SAJ), and the South Florida Water Management District

§Yeh was supported by the U.S. Environmental Protection Agency under grant No. R-82795602 with University of Central Florida.

which is solved with the Eulerian finite element method. The contaminant transport and sediment transport equations, which are solved with the Lagrangian-Eulerian finite element method, are derived based on the mass conservation principle. A parallel in-element particle-tracking algorithm for unsteady flow is applied to backtrack fictitious particles from global nodes to determine the so-called Lagrangian values when the Semi-Lagrangian or the Lagrangian-Eulerian method is used.

This paper addresses the parallelization of such a complex numerical model. In phase I, tasks including data structure and software design, software tool development, as well as tool integration are accomplished. The 2- and 3-D flow modules are expected to be employed in the production stage.

1. INTRODUCTION

WASH123D is a numerical model designed to simulate density-dependent water flow, salinity transport, and sediment and water quality transport in watershed systems. It includes seven modules: 1-D river/stream network module, 2-D overland module, 3-D subsurface module, coupled 1- and 2-D module, coupled 2- and 3-D module, coupled 1- and 3-D module, and coupled 1-, 2-, and 3-D module. It can be used to simulate flows alone, sediment transport alone, water quality transport alone, or flow and sediment and water quality transport simultaneously. When both flow and transport are simulated, the flow fields are computed first. Then the transport is calculated using the computed flow fields at respective times. Salinity-dependent flow is also considered.

In WASH123D, three approaches—the kinematic, diffusive, and dynamic wave—are equipped to characterize the overland flow equations. The kinematic wave equations are solved using the semi-Lagrangian method (backward particle tracking). The diffusion wave equations are approximated with either the Galerkin finite element method or the semi-Lagrangian method. The dynamic equations can first be transformed into the characteristic wave form, which is then solved with the Lagrangian-Eulerian method. The subsurface flow governing equations are discretized with the Galerkin finite element method.

The parallelization of such a complex model starts with the data structure design and then tackles the programming paradigm. The software design targets the four elements—abstraction, encapsulation, modularity, and hierarch—in Booch’s object model [1]. This paper describes how this goal is achieved. The byproduct of this project is the development of the software tool named DBuilder (Domain Builder). The DBuilder serves as a parallel data manager, whose duty includes hiding all the communication, partitioner, and cross-domain coupling. In addition, the software tool named Parallel Particle Tracking (PT) is also linked to perform particle-tracking tasks required by the Lagrangian method.

In this paper, the authors first describe how the Semi-Lagrangian and the Eulerian finite element methods are incorporated in WASH123D. Then code parallelization is stated, followed by demonstrating its scalability in solving the 2-D overland and the 3-D subsurface flow problems on the scalable parallel high performance computers at the Engineer Research and Development Center Major Shared Resource Center (ERDC MSRC) facilitated by the DoD High Performance Computing Modernization Program. The topographic data near the south Miami-Dade County are used to generate the test problem. Performance

measurement is conducted, and the results are analyzed and discussed.

2. MATHEMATICAL FORMULATION

The governing equations of 2-D overland flow and 3-D subsurface flow are described in the following subsections. The numerical methods that this paper presents are those used for demonstration in Section 4.

2.1. Water Flow in 2-D Overland Regime

The governing equations for 2-D overland flow can be derived based on the conservation law of water mass and linear momentum [2] on the physical domain described by the (x, y) coordinate system. This paper is focused on using the diffusive wave model to solve for the overland flow. The continuity equation is written as follows.

$$\frac{\partial h}{\partial t} + \frac{\partial(uh)}{\partial x} + \frac{\partial(vh)}{\partial y} = S + R - E - I , \quad (1)$$

where t is time, the velocity \mathbf{V} is represented by a tuple (u, v) , h is water depth, S is the man-induced source, R is the source due to rainfall, E is the sink caused by evapotranspiration, and I is the sink resulting from infiltration. When the diffusion wave model is considered, the inertia in the momentum equations is assumed not important when compared with others. With the further assumption that eddy viscosity is insignificant as well as the momentum impulse gained from artificial sources/sinks, from rainfall, and that lost to evapotranspiration, to the subsurface media due to infiltration, the velocity \mathbf{V} can be approximated as

$$\mathbf{V} = \frac{-a}{n} \left[\frac{h}{1 + (\nabla z_0)^2} \right]^{2/3} \frac{1}{\sqrt{\| -\nabla H - \frac{h}{2\rho} \nabla(\Delta\rho) + \frac{\tau^s}{\rho gh} \|}} \left(\nabla H + \frac{h}{2\rho} \nabla(\Delta\rho) - \frac{\tau^s}{\rho gh} \right) , \quad (2)$$

where H is water stage. Substituting (2) into (1), the following equation can then be obtained.

$$\frac{\partial H}{\partial t} - \nabla \cdot \left[K \left(\nabla H + \frac{h}{2\rho} \nabla(\Delta\rho) - \frac{\tau^s}{\rho gh} \right) \right] = S + R - E - I , \quad (3)$$

where

$$K = \frac{ah^{5/3}}{n} \frac{1}{[1 + (\nabla z_0)^2]^{2/3}} \frac{1}{\sqrt{\| -\nabla H - \frac{h}{2\rho} \nabla(\Delta\rho) + \frac{\tau^s}{\rho gh} \|}} . \quad (4)$$

To perform transient simulations, either water depth or stage must be given as the initial condition. In addition, appropriate boundary conditions need to be specified to describe the corresponding physical system. WASH123D can have the following three types of boundary conditions: (1) Dirichlet boundary condition with prescribed water depth or stage, (2) flux boundary condition with prescribed flow rate, and (3) water depth-dependent boundary condition with prescribed rating curve.

2.2. Water Flow in 3-D Subsurface Media

The governing equation of subsurface-density-dependent flow through saturated-unsaturated porous media can be derived based on the conservation law or water mass [3]. It is written as follows.

$$\frac{\rho}{\rho_0} F \frac{\partial h}{\partial t} = \nabla \cdot \left[\mathbf{K} \cdot \left(\nabla h + \frac{\rho}{\rho_0} \nabla z \right) \right] + \frac{\rho^*}{\rho_0} q, \quad (5)$$

where ρ is the density of water, ρ_0 is the reference density of water, h is the pressure head, \mathbf{K} is the hydraulic conductivity tensor, z is the potential head, ρ^* is the density of source water, q is the source and/or sink, and F is the water capacity given by

$$F = \alpha' \frac{\theta}{n_e} + \beta' \theta + n_e \frac{dS}{dh}. \quad (6)$$

In the above equation, θ is the moisture content, α' is the modified compressibility of the media, β' is the compressibility of water, n_e is the effective porosity, and S is the degree of saturation. The Darcy's velocity is written as

$$\mathbf{V} = -\mathbf{K} \cdot \left(\frac{\rho_0}{\rho} \nabla h + \nabla z \right). \quad (7)$$

The following boundary conditions can be specified: (1) Dirichlet boundary condition, (2) Neumann boundary condition, (3) Cauchy boundary condition, and (4) variable boundary condition. Details can be found in [4].

3. PARALLEL SOFTWARE TOOL DEVELOPMENT

Parallel distributed-memory applications are an integral part of high performance computing. As more serial applications are converted to the parallel paradigm, algorithm studies and tool development to assist in the process of migrating these applications will be a valuable asset to the application developers. The parallelization of the WASH123D code aims to develop software tools, which are portable and reusable, to benefit the DoD High Performance Computing community. In addition, the programming paradigm is changed to comply with the object-oriented concept. Therefore, the software design focuses on the data structure design, parallel software tool development, and parallel tool integration.

3.1. Data Structure Design

The serial version of the WASH123D code is written in FORTRAN 77. Quite often the debate on the language used for scientific computing leads nowhere, because the performance is determined not only by the adopted language but also by the interaction between the application and the computer system. Since C has better software engineering features than FORTRAN 77 and excellent library support, C is thus the language used to develop the parallel WASH123D. The original serial computational kernel is maintained as what it was to shorten the development time, as there is no parallelization involved and FORTRAN 77 also has its pros. Therefore, the data structure design becomes very important when the goals of object orientation, parallelization, software integration, and language interoperability need to be reached.

To account for problem domains that may include 1-D river/stream network, 2-D overland regime, and 3-D subsurface media, three `WashMesh` objects are constructed. These objects describe the three subdomains, on which a set of partial differential equations (PDEs) is derived to mathematically characterize flow and transport behaviors, within the entire domain. The object `WashDomain`, as sketched in Figure 1, embraces the computational domain. The `WashDomain` also includes a coupling object named `WashCouple`. Moreover, the common phenomena are described by a `WashGlobal` object, and the parallel environmental context is set up by a `WashProcinfo` object. Each subdomain is partitioned, based on its favorite partitioning criteria, to processors by the `DBuilder` developed for the DoD HPC users community. Hence, each `WashMesh` object may include `vtxDomain` and `elementDomain`, which are created and managed by the `DBuilder`, to maintain consistent data structures among processors via ghost vertices/elements on a given mesh. The `WashCouple` may include the `coupler` for (1-D,2-D), (1-D,3-D), and/or (2-D, 3-D). Again, the `coupler` hides all the implementation of a Message Passing Interface (MPI) scheme for communication/synchronization between different dimensions of meshes. The merit of this approach is that the partitioning dependency between meshes can be avoided.

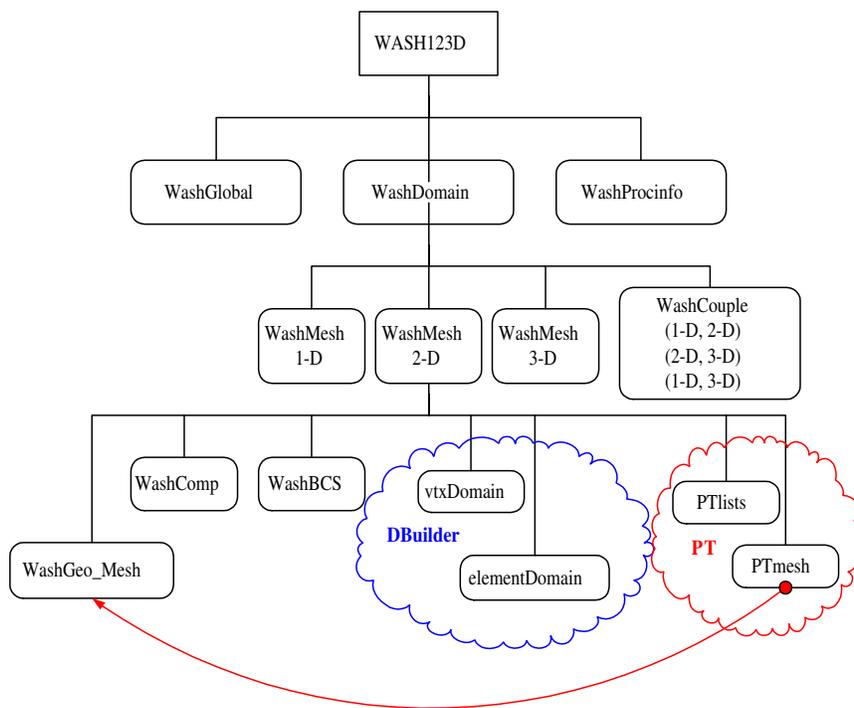


Figure 1. Data structures designed in Wash123D

3.2. Software Tool Development

A majority of scientific applications requires a computational mesh, which is a discretization form of the spatial domain. There are a variety of data that can be associated

with the mesh, represented by sets of vertices, edges, or elements. The associated data perhaps sit on vertices or elements. When the parallelism is employed, the key is to partition the mesh evenly to processors, maintain consistent data among processors, and apply parallel efficient parallel algorithms to reduce communication overhead. DBuilder is thus developed to provide a simple set of Application Programming Interface (API) for users to leverage the work of knowing MPI, graph theorem, and parallel algorithms. In fact, the embedded partitioner in DBuilder is the ParMETIS [5].

Figure 2 shows a section of code implemented in WASH123D to illustrate the use of the DBuilder. DBuilder can build vertex domain, element domain, or both. In this case, both vertex and element domains are built for the finite element method to maintain a balanced number of vertices among processors. The first two function calls pass some geometric information to `DBuild_Init` for building vertex and element domains, respectively. Then, the function `DBuild_Get_file_part` can be called to retrieve the values of local count of entity on the processor. The local count of geometry data is read from a data file. Two times of `DBuild_Domains` are called to build the vertex domain followed by the element domain, whose partition rule is based on the built vertex domain. These two domains have their own local entities and ghost entities, which are owned by other processors. Two steps are required to fill in the element array (i.e., element indices) that is not owned. They are as follows: (1) `DBuild_Set_Type` is called to set the MPI data type for the element array; and (2) `DBuild_Global_update` brings in the values to the array. Based on the built element domain, some vertices may need to live on this processor as a ghost, because one ghost layer is mandatory in the algorithm. `DBuild_Add_ghosts` is developed to achieve this purpose. `DBuild_Set_Type` and `DBuild_Global_update` are then called to bring in the vertex coordinates for the vertex domain. At the bottom, `DBuild_Set_type` is called to set up the data type for updating the data on ghost vertices. Obviously, the data are associated with vertices because the data type is set on the vertex domain. After the domains are built, the rest of parallelization is basically to place `DBuild_Global_update` at appropriate locations in the code.

3.3. PT Tool Integration

With the Picard method dealing with nonlinearity of the 2-D overland flow, the linearized equation can be solved by using particle tracking to compute the total-time-derivative term and by manipulating integration along the tracking path for the source/sink term. Since the dependent variable, either water depth or water stage, can be obtained by solving the linearized equation independently, the nature of this application is perfectly suited for parallel implementation. In this paper, the PT software [6] is facilitated with a new pathline computation kernel to accurately track particles under unsteady flow fields. The design goal of the PT software development is to interface to different software libraries such as Scalable Unstructured Mesh Algorithms and Applications (SUMAA3d)[7] in addition to application codes (e.g., FEMWATER and WASH123D). This goal is achieved through a software architecture specifying a lightweight functional interface. API maintains the full functionality required by particle-mesh methods. To efficiently incorporate different mesh (structured or unstructured) programming environments, the PT software uses an abstract particle-mesh interface (PMI) to interact with the parallel mesh software programming environment. This interface is illustrated in Figure

```

/** Initialization for vtxDoamin and elementDomain */
ierr = DBuild_Init(num_global_vertices, num_ghost_layer, 0,point2neighbor_list,
                  proc_set,&vtx_neighbor_list,&vtx_neighbor_list,
                  &vtx_coord,total_bytes_of_each_vtx, vtxDomain);
ierr = DBuild_Init(num_global_elements, num_ghost_layer,num_neighbors_per_elm,
                  &num_entries_per_elm,proc_set,&element_array,&elm_neighbor_list,
                  NULL,0, elementDomain);

/** Read mesh from a file --- partial file is read on each processor */
ierr = WashRead_geom3(fd, mesh); /** fill in coord and element arrays
                                in mesh and then create neighbor list */

/** Build domains */
ierr = DBuild_Domains(1, NULL, vtxDomain);
ierr = DBuild_Domains(3, vtxDomain, elementDomain);

/** update element_array for ghost elements */
ierr = DBuild_Set_type(num_entries_per_elm*sizeof(int), &dType, elementDomain);
ierr = DBuild_Global_update(element_array,dType,elementDomain);

/** add ghost vertices based on elementDomain */
ierr = DBuild_Add_ghosts(element_array,num_local_vertices,num_neighbors_per_elm,
                        num_entries_per_elm,vtxDomain);

/** bring in the coordinates for ghost vertices */
ierr = DBuild_Set_type(num_dir*sizeof(double), &dType, vtxDomain);
ierr = DBuild_Global_update(coord,dType,vtxDomain);

/** build data types for data gathering or scattering */
ierr = DBuild_Set_type(sizeof(double), &mesh->doubleType, vtxDomain);

```

Figure 2. Code containing DBuilder APIs

3. The PT software architecture [8,9] contains the particle-tracking software implementation, which encapsulates the functionality required by most particle-tracking applications. In this way, details of the parallel implementation are hidden from the application programmer. In addition, a particle API is provided to allow users to specify methods to create, retrieve, or modify particle attributes. The gateway between the PT software and the selected mesh programming environment (e.g., SUMAA3d), which may include an API to increase interoperability, is the abstract PMI that has been developed [8].

To integrate with the PT software tool, a set of PMI and particle API have been built in WASH123D. Most of them require a couple of lines such as

```

PTparticle *PMIget_part_list_from_tri(void *mesh,PTelm *tri)
{ /*@ PMIget_part_list_from_tri - get the particle list from
   the tri.
   @*/
   tri_data *dptr;
   int j = ((int *)tri-((WashGeo_Mesh *)mesh)->ie) / ie_dim1;
   dptr = ((tri_data *)((WashGeo_Mesh *)mesh)->tri_user_data)+j;
   return (PTparticle *) (dptr->pt_list);

```

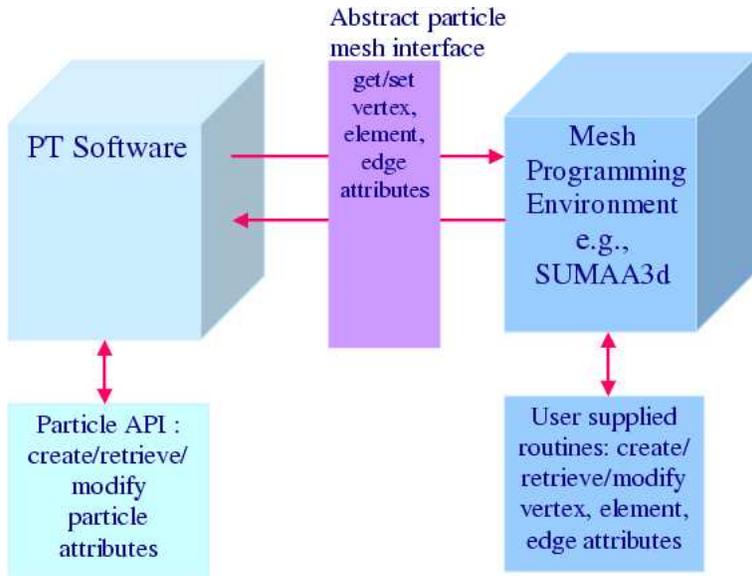


Figure 3. The particle-tracking software architecture and its interface to an existing mesh programming environment

}

The above demonstrates that the effort of developing the well-designed PT software tool not only saves implementation time but also allows for the leverage of this work in the development of the parallel WASH123D code.

4. EXPERIMENTAL RESULTS

The Biscayne Bay and the Florida Bay rely on substantial amounts of distributed fresh water to sustain the estuarine ecosystem. During the past century, field observations suggest that the delivery of fresh water to the bays has changed from overland sheet flow to one controlled by releases of surface water at the mouth of canals. The existing freshwater delivery system is stressful to fish and benthic invertebrates in the bays near the canal outlets. Current restoration efforts in southern Florida are underway for alternative freshwater management plans that could change the quantity, the timing, and quality of freshwater delivery to the bays by restoring coastal wetlands along the western shoreline of the Biscayne Bay and the northeastern shoreline of the Florida Bay.

Topographic data at the south of Homestead, Florida, are used to demonstrate how the parallel computation code is more efficient than a single-processor system in the regional-scale watershed system. Figure 4 shows the location of the application site. Figure 5 shows the colored contour of land surface elevation. The 2-D overland domain, which covers about 530 square miles, is discretized with 31,484 vertices and 62,409 elements. The underlying 3-D domain contains 686,499 elements and 377,808 nodes.

The Manning's roughness was set to 0.02 for the overland flow. The subsurface medium is sandy loam and is assumed isotropic, where the saturated hydraulic conductivity was 1,000 ft/day. The soil retention curves for the unsaturated zone were generated with the

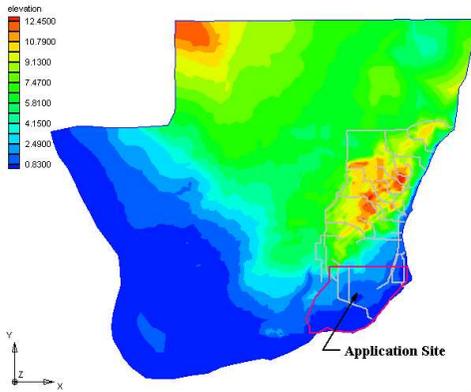


Figure 4. Location of the application site

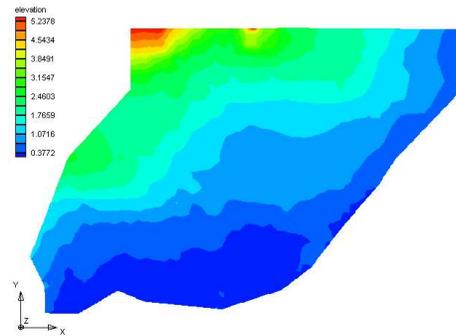


Figure 5. Color-shaded land surface elevation of 2-D overland domain

van Genuchten functions.

In computing 2-D overland flow, on the upstream boundary a time-dependent stage was specified, and a depth-dependent flux (rating curve) was given on the downstream boundary. Figure 6 shows the boundary conditions for 2-D overland flow. As the transient simulation started, a constant rainfall rate of $2.3 \times 10^{-7} ft/s$ was applied for the first 12 hours and no rainfall for the rest of simulation period of 2 days. In performing the 4-day simulation of 3-D subsurface flow, an impermeable boundary condition was assumed for the bottom boundary face; a time-dependent head was specified at the upstream boundary and at the downstream boundary. Figure 7 shows the boundary conditions for 3-D subsurface flow.

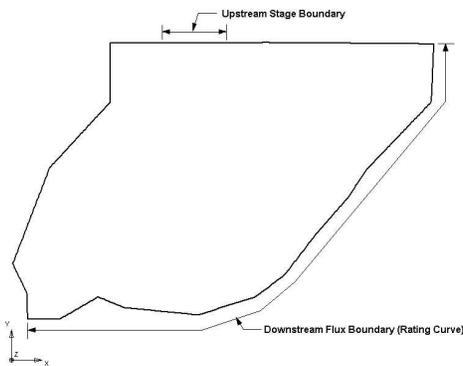


Figure 6. Boundary conditions of 2-D overland flow

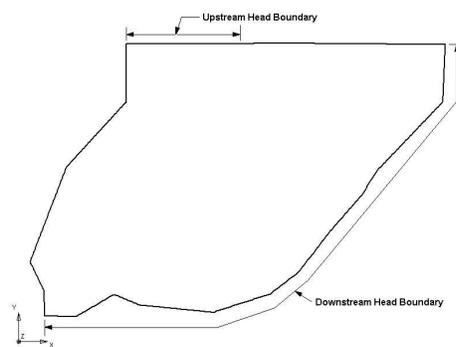


Figure 7. Boundary conditions of 3-D subsurface flow

Figure 8 shows the simulated overland water depth at the time = 48 hours. Figure 9 shows the simulated subsurface pressure head at the end of the third day.

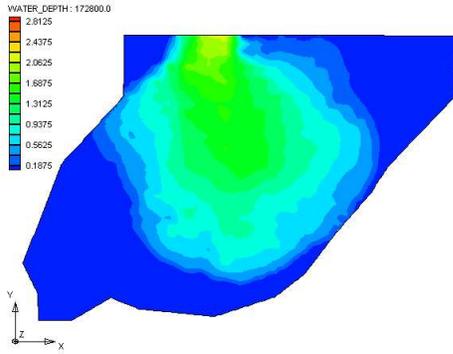


Figure 8. Simulated overland water depth at time = 48 hours

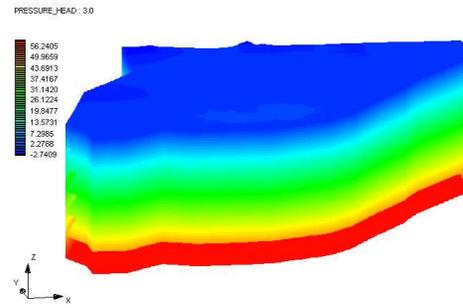


Figure 9. Simulated subsurface pressure head at time = 72 hours

Figure 10 plots the wall clock time vs. number of processors for 2-D overland flow simulation and 3-D subsurface flow simulation. The Compaq AlphaServer SC45 machine is configured with 128 nodes connected by a 64-port, single-rail Quadrics high-speed interconnect switch. Each node contains four 1 GHz Alpha EV 68 processors and four gigabytes of RAM. In addition to the Compaq machines at ERDC MSRC, there are three SGI Origin 3000 machines, and each of them is composed of a shared-memory multiprocessor 512-processor system. One of them is an Origin 3900 system containing 512, 700-MHz R16000 CPUs. Each system contains 512 GB of RAM. From this figure, one can observe that the SC45 outperforms the Origin when the simulation runs on less than 128 processors. As for the 128-processor simulation, the parallel efficiency is around 50 percent on the Origin, listed in Table 2, and less than 30 percent on the SC45 (see Table 1). Therefore, the Origin has better scalability than the SC45. The cause can be the interconnection switch configuration and the file system because the simulation involves I/O access and communication. For the 3-D simulation, the parallel efficiency drops to 20 percent, shown in Table 3, when 128 processors are used. Obviously, one can observe from this table that the bottleneck is set at 32 processors. Further investigation of parallel solver efficiency is demanded. More sophisticated parallel solvers are currently integrated to DBuilder.

5. CONCLUSION AND FUTURE WORK

The software tool DBuilder has successfully embedded MPI implementation so that application users do not need to know the MPI library and parallel algorithms. The following tasks for the CHSSI project have been completed: implementation of dynamic memory allocation, DBuilder functionality enrichment, parallel PT software integration, and the parallel performance evaluation. From the experimental demonstration, verification and speedup have been investigated. Further improvement of the parallel linear solver is an ongoing task to break the bottleneck at 32 processors. The WASH123D team is pushing forward the development of the coupling module as well. Large-scale field problems are also desired for a large number of processors.

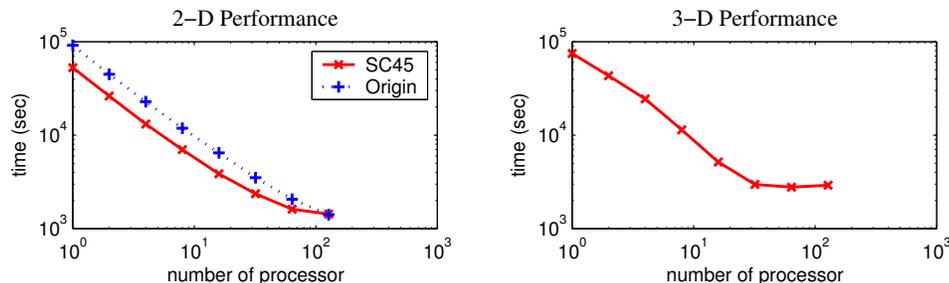


Figure 10. Wall clock time for 2-D overland flow simulation (left) and for 3-D subsurface flow simulation (right)

Table 1

Scalability on the Compaq SC45: timings for 2-D overland flow simulation

Number of processors	Time (sec)	Speedup	Parallel efficiency
1	52628.88	-	-
2	26259.75	2.004	1.002
4	13173.48	3.995	0.999
8	6984.73	7.535	0.942
16	3866.91	13.610	0.851
32	2356.91	22.350	0.698
64	1611.9	32.650	0.510
128	1429.63	36.813	0.288

REFERENCES

1. Grady Booch. *Object-oriented analysis and design with applications (2nd ed.)*. Benjamin-Cummings Publishing Co., Inc., 1994.
2. J. D. Wang and J. J. Connor. Mathematical modeling of near coastal circulation. Technical Report Report MITSG 75-13, Massachusetts Institute of Technology, MA, 1975.
3. H. J. Lin, D. R. Richards, C. A. Talbot, 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. Technical Report CHL-97-12, Engineer Research and Development Center, U.S. Army Corps of Engineers, MS, July 1997.
4. G.-T. Yeh, H.-P. Cheng, G. Huang, F. Zhang, H.-C. Lin, E. Edris, and D. Richards. A numerical model of flow, heat transfer, and salinity, sediment, and water quality transport in watershed systems of 1-d stream-river network, 2-d overland regime, and 3-d subsurface media (WASH123D: Version 2.0). Technical Report CHL-03-XX, Engineer Research and Development Center, U.S. Army Corps of Engineers, MS, 2003.

Table 2
Scalability on the SGI Origin 3900: timings for 2-D overland flow simulation

Number of processors	Time (sec)	Speedup	Parallel efficiency
1	91341.05	-	-
2	44859.19	2.036	1.018
4	22804.61	4.005	1.001
8	11839.87	7.715	0.964
16	6471.57	14.114	0.882
32	3509.32	26.028	0.813
64	2062.01	44.297	0.692
128	1409.74	64.793	0.506

Table 3
Scalability on the Compaq SC45: timings for 3-D subsurface flow simulation

Number of processors	Time (sec)	Speedup	Parallel efficiency
1	74818	-	-
2	43205.96	1.732	0.866
4	24436.88	3.062	0.765
8	11380.01	6.575	0.822
16	5145.91	14.539	0.909
32	2969.96	25.192	0.787
64	2775.96	26.952	0.421
128	2916.71	25.652	0.200

5. Research Team led by George Karypis. ParMETIS parallel graph partitioning. <http://www-users.cs.umn.edu/karypis/metis/parmetis/>.
6. J.-R. C. Cheng and P. E. Plassmann. Parallel particle tracking framework for applications in scientific computing. *Journal of Supercomputing*, to appear.
7. Lori Freitag, Mark Jones, Carl Ollivier-Gooch, and Paul Plassmann. SUMAA3d Web page. <http://www.mcs.anl.gov/sumaa3d/>, Mathematics and Computer Science Division, Argonne National Laboratory, 1997.
8. J-R. C. Cheng. *Parallel Particle Tracking Algorithms and Software for Applications in Scientific Computing*. PhD thesis, Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA, USA, 2002.
9. J-R. C. Cheng and P.E. Plassmann. A software architecture for parallel particle tracking algorithms. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDATA'03)*, pages 656–661, 2003.