A Many-Task Parallel Approach for Multiscale Simulations of Subsurface Flow and Reactive Transport

Timothy D. Scheibe, Xiaofan Yang, Karen Schuchardt, Khushbu Agarwal, Jared Chase, Bruce Palmer, and Alexandre Tartakovsky
Pacific Northwest National Laboratory
902 Battelle Blvd
Richland, WA 99354
+1(509)371-7633
tim.scheibe@pnnl.gov

ABSTRACT
Continuum-scale models have long been used to study subsurface flow, transport, and reactions but lack the ability to resolve processes that are governed by pore-scale mixing. Recently, pore-scale models, which explicitly resolve individual pores and soil grains, have been developed to more accurately model pore-scale phenomena, particularly reaction processes that are controlled by local mixing. However, pore-scale models are prohibitively expensive for modeling application-scale domains. This motivates the use of a hybrid multiscale approach in which continuum- and pore-scale codes are coupled either hierarchically or concurrently within an overall simulation domain (time and space). This approach is naturally suited to an adaptive, loosely-coupled many-task methodology with three potential levels of concurrency. Each individual code (pore- and continuum-scale) can be implemented in parallel; multiple semi-independent instances of the pore-scale code are required at each time step providing a second level of concurrency; and Monte Carlo simulations of the overall system to represent uncertainty in material property distributions provide a third level of concurrency. We have developed a hybrid multiscale model of a mixing-controlled reaction in a porous medium wherein the reaction occurs only over a limited portion of the domain. Loose, minimally-invasive coupling of pre-existing parallel continuum- and pore-scale codes has been accomplished by an adaptive script-based workflow implemented in the Swift workflow system. We describe here the methods used to create the model system, adaptively control multiple coupled instances of pore- and continuum-scale simulations, and maximize the scalability of the overall system. We present results of numerical experiments conducted on NERSC supercomputing systems; our results demonstrate that loose many-task coupling provides a scalable solution for multiscale subsurface simulations with minimal overhead.

Categories and Subject Descriptors
J.2 [Computer Applications]: Physical Sciences and Engineering; D.4 [Operating Systems]: D.4.1 [Process Management]: Multiprocessing/multiprogramming/multitasking

General Terms
Algorithms, Management, Performance, Verification.

Keywords
Workflow, scalability, coupling, multiscale simulations

7th Workshop on MTAGS, November 16, 2014, New Orleans, LA, USA.

1. INTRODUCTION
Contemporary subsurface environmental and energy applications involve coupling of hydrologic and biogeochemical processes in the context of a highly heterogeneous environment. The immense disparity in spatial scales between fundamental process scales and application scales poses a severe challenge to predictive modeling and has motivated the study of novel hybrid multiscale modeling approaches. [1]. Continuum-scale models have been used to study subsurface fluid flow, transport, and reactions for many years. These models treat complex porous systems as an effective continuum, with macroscopic properties such as porosity and permeability. They simulate reactions based on concentrations of reactants and products defined over volumes corresponding to elements of the model discretization, and thus assume complete mixing below the resolution of the grid (an assumption that is usually invalid). In contrast, pore-scale models explicitly represent individual soil grains and pore spaces, and are discretized at scales at which diffusion is relatively fast, thus rendering sub-grid mixing a much better assumption. As a result, pore-scale models can more accurately model mixing-controlled processes such as mineral precipitation [2]. Kinetic models and genome-scale microbial models are being developed to improve our understanding of surface reactions that cannot be fully captured by pore-scale models. Since even a small domain may contain on the order of billions of particles, or trillions of microbes or molecules, it is prohibitively expensive to model entire domains using the fine grained methods. Thus, hybrid multiscale models [1] are being developed to incorporate pore-scale models within limited sub-domains of traditional models. Hybrid multiscale models are good candidates for a many-task parallel workflow, because within a single simulation domain there may be many sub-domains (adaptively defined) within which pore-scale simulations are desired. Each of these simulations can be executed using a parallel code (e.g., using domain decomposition methods), and many such simulations can be performed in parallel (often in a fully decoupled mode to eliminate communications). A third level of concurrency can be obtained in the case where multiple realizations of the same model

are needed, for example in Monte Carlo simulations for uncertainty quantification. We describe an example of such a model in this paper.

At each scale in a hybrid multiscale simulation, there is a choice of alternative simulators that can be applied with the selection of a particular code dictated by the unique features of both the problem definition and the simulators. In the prototype model proposed by Scheibe et al. [3] and described in this paper, a Swift workflow couples the STOMP [4] continuum code (macroscale) with the SPH [5] pore-scale code (microscale). Swift [6, 7] provides a data flow framework for controlling execution and exchanging data across workflow steps via files. The other major function of a coupling framework, namely data transformation, is accomplished via custom scripts integrated into the workflow. Other components of the workflow perform the following functions: dynamically determine the number and locations of pore-scale domains; transform continuum data to pore scale and vice versa, intelligently schedule the pore scale simulations.

The numerical experiment setup and a high level view of the workflow are shown in Figure 1. In this iterative process, the macro-scale model (STOMP) first executes a single time step over the entire domain. The output is processed to evaluate adaptivity criteria that determine the location and number of micro-scale models and to produce the required input files for the micro-scale simulations (performed by a pore generator script - PG). The MPI-based SPH simulations are scheduled concurrently depending on the number of processors available and the performance profile of the code. The workflow hence follows the *many task computing* paradigm [8, 9], executing multiple parallel tasks concurrently. In our test problem, all micro-scale simulations occur in a single vertical column of macro-scale grid cells, and up to 60 such simulations can be performed within a single time step (one loop of the workflow). Results of micro-scale simulation are collected and processed to provide effective reaction rates and concentrations as feedback to the macro-scale model through a grid parameter generator script (GPG) and starting the next iteration of the loop. All process communications are performed via files already used and produced by the codes.

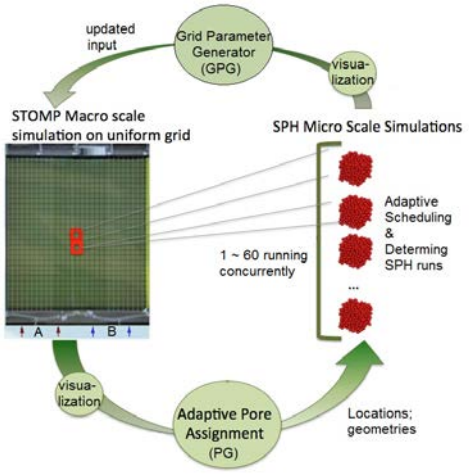

Figure 1. The coupled workflow and iterative process

While a tightly-coupled model may offer performance benefits, we chose to follow a loose coupling approach for some important reasons. "First, the two models have very different data structures; tight integration of shared structure is not required. Second, each of these codes is being developed independently by separate groups and undergoing large-scale development. In addition, the code which determines pore regions, their locations and characteristics would also have to be integrated. Tight integration will require significant effort to manage and be disruptive to ongoing efforts [10]." Third, we wanted to employ a framework that could be readily adapted to plug in alternative micro-scale and macro-scale simulators as needed. Finally, adding analysis and visualization methods to the coupled process further complicates the model. This loose coupling approach was largely non-invasive to the two simulators (SPH and STOMP), with minimal changes required to the SPH code to implement model boundary conditions that allowed the individual micro-scale simulations to be fully decoupled from each other.

2. BACKGROUND

2.1 STOMP and SPH

The continuum-scale simulation is performed using the Subsurface Transport Over Multiple Phases (STOMP) simulator [4]. Here we are considering only saturated flow and solute transport with aqueous-phase reactions in 2D on a modest number of grid nodes, and therefore we use the serial water-only mode with reactions (ECKEChem module).

The Smoothed Particle Hydrodynamics (SPH) method is a fully lagrangian mesh-free particle-based method, which is particularly well suited to simulate problems that involve moving interfaces and dynamic pore geometry. We use an in-house scalable parallel implementation of SPH [5] as the pore-scale simulator.

2.2 Swift

A Swift script describes data, application components, and invocations of applications. A Swift workflow generally involves executing a large number of independent tasks in an HPC or distributed environment. The advantages to using Swift include an elegance of remote execution that is inherent to the language. Swift also provides file and data management capabilities.

Swift has an inherent parallel nature; when iterating over arrays, each task is performed in parallel. This makes the execution of parameter studies much more efficient. Also, the complexities of parallelization are encapsulated. This makes the launching of multiple remote jobs in parallel and monitoring their status simple to implement. Swift will launch every job in parallel and wait for them to finish. Once each job is completed, Swift will check for the expected output files.

3. RELATED WORK

There are many approaches that can be considered for coupling multiscale components into a single hybrid model. One approach that has been quite successful and provides for good performance is the use of an application-specific, MPI-based coupler component. This method has been applied in climate simulation [11], gas turbine applications [12] and other domains [13-15]. This approach works best when the set of components are fairly static and data transformations are well defined. The main drawback is that the coupler may be intrusive to each component and costly to implement. Frameworks that generalize the coupling concept while still providing for a single MPI-based code have also been developed such as [16] for numerical relativity applications and the Common Component Architecture (CCA) [17] as a general framework. This approach also requires

significant design change to existing codes and in practice has not been readily adapted to other hybrid modeling efforts.

Workflow based approaches using scripting languages have been popular due to the ease of implementation, maintenance and portability. A coupled model using python as a scripting language was developed for multi-physics simulations [12]. The ESSE used shell scripting to develop a workflow for running an ensemble of climate model simulations [15]. A number of formal workflow frameworks [eg.,. 18, 19] have been implemented over last few years that provide an abstraction from the details of workflow execution, job scheduling, resource management and error handling etc. The Swift workflow language, used to develop the hybrid subsurface model, offers an implicitly parallel and deterministic programming model [7], which is central to our multi-parallel task based workflow design. It also provides functional mappers, which allows external applications to be applied to file collections. Moreover, a C-like syntax and abstraction from complex details of parallel execution greatly simplifies the implementation process. The IPS (Integrated Plasma Simulator) framework [13] used Swift for coupled multi-physics simulation of fusion plasmas. For loosely-coupled approaches, communication is performed via files, potentially introducing a significant bottleneck. However, the use of files fits naturally into the components and can be optimized, to some extent, using ram-disk [22], where available, or approaches such as HDF5 virtual file drivers [23] that employ the underlying mechanism of the HDF5 API to use memory rather than files for storage.

4. DESIGN AND IMPLEMENTATION

4.1 Bimolecular Reaction Experiment

Our initial numerical experiment (Figure 1) simulates the parallel transport of two solutes with an irreversible mixing-controlled kinetic reaction occurring at the interface between the two solutes, generating a third solute. The system is filled with a homogeneous porous medium (sand). The sand is saturated with water, and two solutes (denoted as A and B) are injected at the bottom and flow to the top at a specified rate. As the solutions flow upward through the flow cell, they mix along the centerline, leading to reaction and formation of the third solute (C). The rate of reaction at the interface is strongly controlled by the rate of lateral diffusion of the two reactants. The mathematical approach for coupling the pore- and continuum-scale simulations is described in [20].

Our numerical experiment is performed on a 2D system (30.5 cm x 30 cm, Figure 2). The macroscale (STOMP) simulations use a regular mesh of size 61 x 60 cells. Each SPH geometry is homogenous with a size of 0.5 x 0.5 cm (corresponding to a single STOMP cell) and containing 40,000 particles. STOMP executes on a single processor while the SPH code executes on the "best" number of available processors. The mechanism for this is described in the Adaptive Scheduling section below. The system is modeled after the mixing-controlled reaction experiment reported in [21], with the primary difference being that we consider for simplicity a homogeneous (aqueous-phase only) reaction rather than a mineral precipitation reaction. A specified flux boundary condition is applied at the bottom of the domain, with a Darcy velocity of 1 cm/min, and a specified pressure is imposed at the top of the domain simulating the free outflow boundary of the experiment. No-flow conditions are specified at right and left boundaries.

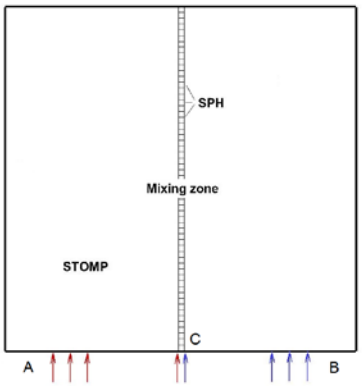

Figure 2. Computational domain.

4.2 Swift Workflow

Our hybrid subsurface model workflow is implemented using the Swift workflow language [6, 7]. Swift's ability to maintain relationships over multiple iterations between different components of the workflow provides a powerful interface to track and maintain each piece of data as it is generated during the course of the simulation. Also, the generation of a virtual environment for executing subtasks and ensuing garbage collection ensures that the temporary data generated over the course of the run is safely discarded, with only the necessary data being preserved for post-processing.

As codes execute in temporary virtual environments, ensuring all input files generated from multiple sources be present in the virtual directory is cumbersome and prone to errors. Also, as with most of the scientific codes, STOMP and SPH generate multiple output files, each marked with unique time step. Specifying an arbitrary number of files as an output of a code is not fully supported in Swift. This forces us to do a 'tar' on all output files to create single file, which ensures that all files are preserved. The files marked with the last time step also end up serving as input to other components of the workflow. Since, the name of the output file with last time step is unknown until runtime, Swift cannot be expected to bring back that file. To overcome this limitation, a simple bash code, identifies the file with latest time step and moves it to a pre-specified name, which is provided to Swift. Another requirement of our Swift workflow is a process to specify the number of SPH runs. The PG component calculates the number of SPH runs in a particular iteration and creates input files for each SPH simulation. Candidate SPH domains become activated when the initial concentrations of reactants reach a user-specified minimum threshold. Swift, however, needs to know the number of files that will be produced as output of PG, so it can wait for all of them to get produced. To address this issue, we split our PG algorithm into two sub-steps. The first step calculates number of pore-regions '*n*' required in a particular iteration and hands this information back to Swift workflow. Swift's fixed_array_mapper is then used to specify files for each of the '*n*' SPH simulations, produced by PG.

Our Swift workflow consists of an app each for STOMP, PG_calc_num, PG_createfiles, SPH and GPG. Limited modifications were made to the pre-existing STOMP and SPH simulators to facilitate file-based exchange of boundary conditions and eliminate need for direct communications, but these modifications were minimal in nature. A foreach construct is used to run all SPH simulations in parallel, as per our adaptive

scheduling policy, described in the next section. A hybrid_model function consists of all these app components, and defines a single iteration of the workflow. An iterative loop over the hybrid_model function is used, enforcing serial execution between iterations, where outputs from one iteration serve as input to the next iteration. A maximum number of iterations is specified at the command line by the user. Swift is configured to run locally on the system and definitions are provided (path to code executables) for each of the "apps" in the workflow.

4.3 Adaptive Scheduling

The pore-scale simulations in the hybrid subsurface model are the most expensive part of the workflow. The number of these simulations changes at each iteration as solute concentrations move throughout the system. The pore-scale simulations are launched using the Swift's "foreach" construct which executes the tasks in parallel. Each of these SPH runs is launched as parallel job. Our adaptive scheduling algorithm focuses on minimizing the run time for running these multiple SPH simulations, based on the number of processes available. Some of the key features are:

1) Define an optimal range for number of processes to be used for each SPH run: We observed that the scalablity of the SPH simulations was limited due to the relatively small number of particles used in each simulation. To determine the optimal number of processes required for SPH runs, we performed multiple scaling tests, varying the number of processes for three candidate particle discretizations (Np=40,000, 80,000 and 160,000). SPH run time decreases with increasing number of processors initially, but eventually the run time flattens and then increases as additional processors are added, because communication becomes the predominant factor in runtime as opposed to computation. We also observed that a minimum number of processors is required by SPH to provide sufficient memory. Hence we define a range for number of processes (minprocs, maxprocs) suitable for executing a single SPH run. For Np=40,000, which was deemed sufficient to obtain an accurate solution, minprocs=1152 (48 nodes * 24 cores/node) and maxprocs=2304 (96 nodes * 24 cores/node) (Figure 3).
2) Schedule multiple batches of simultaneous SPH simulations: The scheduling algorithm determines if there are enough nodes in the runtime allocation to run all SPH tasks together. If there are enough nodes, then all SPH tasks can be scheduled together. In this case, the algorithm divides the resources equally amongst all SPH tasks, ensuring that each does not use more than "maxprocs". If the number of nodes is not sufficient to run all SPH tasks together, the runs are done in multiple batches in an iterative manner using "minprocs" for each SPH. In the worst case scenario, all SPH simulations can be launched as serial jobs in an iterative manner.
3) Map from processes to nodes based on system configuration: To avoid wasting node computation power and avoid scheduling issues across node boundaries, all data regarding available number of processes, minprocs and maxprocs is converted to number of nodes (by dividing 24 procs/node) in the input and configuration files and the scheduling algorithm is applied accordingly.

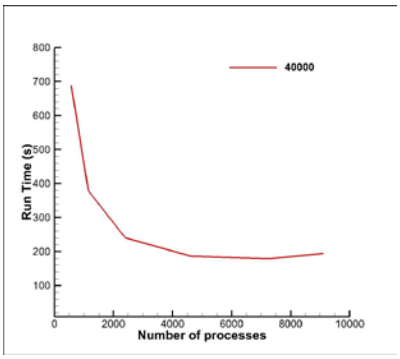

Figure 3. Scaling tests for SPH run times, Np=40000

4.4 Eliminating SPH Runs

During the course of our hybrid model runs, it was observed that the SPH simulations tend toward a quasi-steady condition over time. Initially the boundary conditions lead to rapid influx of reactants and an increase in the reaction rate as mixing proceeds, but after some relatively small number of iterations the influx of the reactants is balanced by the efflux plus consumption by the reaction, and the system becomes stable. Since the effective reaction rate (passed to the macro-scale model) is no longer changing, it is not necessary to perform pore-scale simulations in future iterations.

To save critical time and resources, we incorporate methods in workflow to adaptively turn off execution for those SPH cells that have reached steady state. This greatly reduces the run time of the hybrid system, from running potentially running average of ~30 SPH each iteration to just running a few. In our example case it was observed that running only about 15-25 SPH's every iteration was sufficient to model active parts of the complete hybrid model domain.

4.5 Visualization

STOMP results are visualized as 2D spatial plots of concentrations of reactants A, B, and product C (a single time snapshot is shown in Figure 6). When the workflow begins, concentrations of A and B—enter the system at the bottom of the cell and react to form C, shown in the third plot. Values for concentration are indicated using a spectrum of color values. As the workflow continues execution, the concentrations of the three constituents rise into the upper regions of the cell.

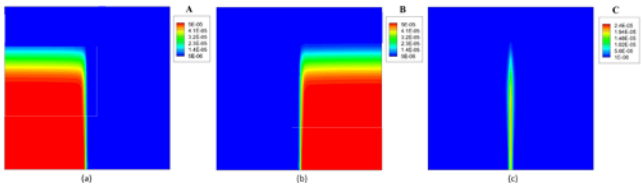

Figure 6. STOMP visualization

Each SPH task is configured to run for 1330 steps (time step = 0.00075), simulating 1 second of time for each iteration of the workflow. Every 100 steps the concentrations are exported to an h5part file (13 files per workflow iteration). Unfortunately, it takes two to three times as long to plot the results from SPH than it takes to execute the simulation. For this reason we removed the automated generation of these plots from the workflow. Instead, we wrote a postprocessing script to generate plots for runs of interest instead of generating all plots during workflow execution.

Three pseudo color plots are produced visualizing concentrations of reactants A, B, and product C (Figure 7, analogous to the STOMP plots in Figure 6 but within a limited pore-scale domain). The circular blue regions indicate areas occupied by solid grains. Only liquid regions are used to plot the concentrations for each constituent. In Figure 7, A and B (left two images, respectively) have mostly reacted at the particle level to form C (right image).

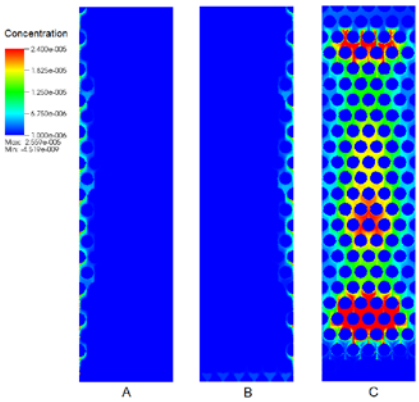

Figure 7. SPH visualization

5. EXPERIMENTAL RESULTS

The entire simulation consists of ~800 iterations, involving less than 60 pore-scale simulations within each iteration, and is run on 1536 nodes (24 processors each) on a Cray XE6 system (Hopper, NERSC). It takes ~96 hr as wall clock time to finish the simulation. 85% of run time is used for SPH runs and less than 10% is used for PG. Currently I/O is not a problem for this 2D experiment. The visualization process is not included in the workflow to save computational time.

6. DISCUSSION

6.1 Overall Summary

We have demonstrated a many-task approach to hybrid multiscale coupling of pore- and continuum-scale porous media flow and reactive transport simulators. The hybrid multiscale approach is relatively new in subsurface hydrology [1], and is well-suited to the use of high-performance computing and a task parallel script-based simulation environment. Loose coupling of many micro-scale tasks within a macro-scale domain was supported by use of the Swift workflow environment, and provided a feasible solution approach to a complex simulation problem. In the case considered, in which pore-scale mixing is a dominant process that cannot be adequately represented in a continuum-only model, the hybrid method provides a new alternative to increase solution accuracy while maintaining computational feasibility (relative to simulating pore scale processes over the entire spatial and temporal domain). The model uses Swift's data mapping and management, error handling and inherent parallel model execution capabilities and also includes post-processing and visualization. The 2D mixing-controlled example problem serves as a test case to demonstrate the capability and accuracy of the current hybrid multiscale model, which can be extended to more realistic problems.

6.2 Swift Related Issues and Appropriateness

When using Swift we have encountered a few challenges that we have had to work around. First is a documented issue of how Swift handles file collections when the number of files is unknown. Swift's execution approach is to run all things in parallel. Execution is blocked if the input from one process depends on the output from another process. Swift needs to know how many files to wait for until it can stop blocking. Our output files have a known naming scheme but the number of output files can vary. There doesn't seem to be an easy way for Swift to handle a variable number of these files. To get around this, we archive all the output files (into one single "tar" file) after the simulation is complete. This way, Swift blocks execution until the single archive file is produced. Downstream processes that need these files are then responsible for unpacking the files before using them.

Another issue encountered with Swift involves input file staging. Before running a simulation, Swift creates a new job directory where it generates symbolic links to each of the input files. If the input file is located inside a subdirectory created by a previous job, a new directory is created to mirror its relative location inside the job directory. When the simulation is invoked, the input files are all located inside subdirectories instead of inside the base run directory. To work around this we moved the files using a shell script into the base run directory before invoking the simulation.

A third issue when working with Swift was dynamic assignment of variables. In Swift, when a variable is assigned a value, the value is final. This reduces the language flexibility when creating logic to swap I/O files during workflow execution based on outside information. Arrays in Swift can have multiple values assigned, so this approach was used to develop the desired logic.

Swift also simplifies data management by implicitly removing files that are not specified as part of the workflow. However all files that might be needed for provenance/visualization or other data analysis capabilities are identified in the workflow in order to be preserved. We also would like to mention that our Swift workflow is portable but does not use the recently developed JETS [7] in which the tasks are managed by an MPICH based task manager.

7. ACKNOWLEDGMENTS

This research was supported by the U. S. Department of Energy (DOE) offices of Biological and Environmental Research (BER) and Advanced Scientific Computing Research (ASCR) under the Scientific Discovery through Advanced Computing (SciDAC) program and the PNNL Subsurface Biogeochemical Research Scientific Focus Area (SFA) project. Computations described here were performed using computational facilities of the National Energy Research Scientific Computing Center (NERSC), a national scientific user facility sponsored by DOE Office of Science. PNNL is operated for the DOE by Battelle Memorial Institute under Contract No. DE-AC06-76RLO 1830.

8. REFERENCES

[1] Scheibe, T. D., Murphy, E. M., Chen, X., Carroll, K. C., Rice, A. K., Palmer, B. J., Tartakovsky, A. M., Battiato, I., and Wood, B. D. 2014. An analysis platform for multiscale hydrogeologic modeling with emphasis on hybrid multiscale methods. *Ground Water*, published online March 13. DOI=http://dx.doi.org/10.1111/gwat.12179.

[2] Tartakovsky, A. M., Meakin, P., Scheibe, T. D., and Wood, B. D. 2007. A smoothed particle hydrodynamics model for reactive transport and mineral precipitation in porous and

fractured porous media. *Water Resour. Res.* 43(5): No. W05437. DOI=http://dx.doi.org/10.1029/2005wr004770.

[3] Scheibe, T. D., Tartakovsky, A. M., Tartakovsky, D. M., Redden, G. D., and Meakin, P. 2007. Hybrid numerical methods for multiscale simulations of subsurface biogeochemical processes. In SciDac 2007: Scientific Discovery Through Advanced Computing, U487-U491. *J. Phys.: Conference Series* 78: 012063. DOI=http://dx.doi.org/10.1088/1742-6596/78/1/012063.

[4] Nichols, W. E., Aimo, N. J., Oostrom, M., and White, M. D. 1997. *STOMP Subsurface Transport Over Multiple Phases: Application Guide PNNL-11216 (UC-2010)*, Pacific Northwest National Laboratory.

[5] Palmer, B., Gurumoorthi, V., Tartakovsky, A. M., and Scheibe, T. D. 2010. A component-based framework for smoothed particle hydrodynamics simulations of reactive fluid flow in porous media. *Int. J. High Perform. C.* 24, 228-239. DOI=http://dx.doi.org/10.1177/1094342009358415.

[6] Wilde, M., Foster, I., Iskra, K., Beckman, P., Zhang, Z., Espinosa, A., Hategan, M., Clifford, B., and Raicu, I. 2009. Parallel scripting for applications at the petascale and beyond. *Computer* 42, 50-60. DOI=http://dx.doi.org/10.1109/MC.2009.365.

[7] Wilde, M., Hategan M., Wozniak J. M., Z., Clifford B., Katz D. S., Foster I. 2011. Swift: A language for distributed parallel scripting. *Parallel Comput.* 27(9), 633-652. DOI=http://dx.doi.org/10.1016/j.parco.2011.05.005.

[8] Raicu, I., Foster, I., and Zhao, Y. 2008. Many-task computing for grids and supercomputer. *IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS)*, November, 2008.

[9] Ogasawara, E., de Oliveira, D., Chirigati, F., Barbosa, C. E., Elias, R., Braganholo, V., Coutinho, A., and Mattoso, M. 2009. Exploring many task computing in scientific workflows. *IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS)*, November, 2009.

[10] Schuchardt, K. L., Palmer, B., Agarwal, K., and Scheibe, T. D. 2011. *Many Parallel Task Computing for a Hybrid Subsurface Model*. SciDAC 2011.

[11] Collins, W. D., Bitz, C. M., Blackmon, M. L., Bonan, G. B., Bretherton, C. S., Carton, J. A., Chang, P., Doney, S. C., Hack, J. J., Henderson, T. B., Kiehl, J. T., Large, W. G., Mckenna, W. G., Santer, B. D., and Smith, R. D. 2006. The Community Climate System Model version 3 (CCSM3). *J. Climate* 19, 2122-2143. DOI=http://dx.doi.org/10.1175/jcli3761.1.

[12] Schlüter, J. U., Wu, X., Kim, S., Shankaran, S., Alonso, J. J., and Pitsch, H. 2005. A framework for coupling Reynolds-Averaged with Large Eddy Simulations for gas turbine applications. *J. Fluid Eng.-T. ASME* 127(4):608-615. DOI=http://dx.doi.org/10.1115/1.1994877.

[13] Foley, S. S., Elwasif, W. R., Bernholdt, D. E., Shet, A. G., and Bramley, R. 2010. Many task applications in the integrated plasma simulator. *IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS)*, November, 2010.

[14] Marshall, P., Woitaszek, M., Tufo, H. M., Knight, R., McDonald, D., and Goodrich, J. 2009. Ensemble dispatching on an IBM Blue Gene/L for a bioinformatics knowledge environment. *IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS)*, November, 2009.

[15] Evangelinos, C., Lermusiaux, P. F., Xu, J., Haley, P. J., and Hill, C. N. 2010. Many task computing for multidisciplinary ocean sciences: real-time uncertainty prediction and data assimilation. *IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS)*, November, 2010.

[16] Allen, G., Dramlitsch, T., Foster, I., Karonis, N. T., Ripeanu, M., Seidel, E., and Toonen, B. 2001. Supporting efficient execution in heterogeneous distributed computing environments with cactus and globus. *ACM Supercomputing Conference*, November 2011, Denver, CO. DOI=http://dx.doi.org/10.1109/SC.2001.10007.

[17] Allan, B. A., Armstrong, R., Bernholdt, D. E., Bertrand, F., Chiu, K., Dahlgren, T. L., Damevski, K., Elwasif, W. R., Epperly, T. G. W., Govindaraju, M., Katz, D. S., Kohl, J. A., Krishnan, M., Kumfert, G., Larson, J. W., Lefantzi, S., Lewis, M. J., Malony, A. D., McInnes, L. C., Neiplocha, J., Norris, B., Parker, S. G., Ray, J., Shende, S., Windus, T. L., and Zhou, S. J. 2006. A component architecture for high-performance scientific computing. *Int. J. High Perform. C.* 20 (2), 163-202. DOI=http://dx.doi.org/10.1177/1094342006064488.

[18] Costa, R., Brasileiro, F., Filho, G. L., and Sousa, D. M. 2009. OddCI: Ondemand distributed computing infrastructure. *IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS)*, November, 2009.

[19] Raicu, I., Foster, I., Wilde, M., Zhang, Z., Iskra, K., Beckman, P., Zhao, Y., Szalay, A., Choudhary, A., Little, P., Moretti, C., Chaudhary, A., and Thain, D. 2010. Middleware support for many-task computing. *Cluster Comput.* 13, 291-314. DOI=http://dx.doi.org/10.1007/s10586-010-0132-9.

[20] Tartakovsky, A. M. and Scheibe, T. D. 2011. Dimension reduction method for advection-diffusion-reaction systems. *Adv. Water Resour.* 34(12), 1616-1626. DOI=http://dx.doi.org/10.1016/j.advwatres.2011.07.011.

[21] Tartakovsky, A. M., Redden, G., Lichtner, P. C., Scheibe, T. D., and Meakin, P. 2008. Mixing-induced precipitation: Experimental study and multiscale numerical analysis. *Water Resour. Res.* 44(6): No. W06s04. DOI=http://dx.doi.org/10.1029/2006wr005725.

[22] Wickberg, T. and Carothers, C. 2012. The RAMDISK storage accelerator: a method of accelerating I/O performance on HPC systems using RAMDISKs. *In proceedings of the 2nd International Workshop on Runtime and Operating Systems for Supercomputers (ROSS '12)*. ACM, New York, NY, USA, Article 5. DOI=http://dx.doi.org/10.1145/2318916.2318922.

[23] Biddiscombe, J., Soumagne, J., Oger, G., Guibert, D., and Piccinali, J. G. 2011. Parallel computational steering and analysis for HPC applications using a paraview interface and the HDF5 DSM virtual file driver. *In Proceedings of the 11th Eurographics conference on Parallel Graphics and Visualization*, Aire-la-ville, Switzerland, 2386244, 91-100. DOI=http://dx.doi.org/10.2312/egpgv/egpgv11/091-100