Cloud computing with ParFlow: Introduction of a newly developed Web interface

Stefan Kollet\textsuperscript{1}, Jens Schumacher\textsuperscript{2}, Claudius Bürger\textsuperscript{3}, Detlef Bösel\textsuperscript{4}

\textsuperscript{1}Meteorological Institute, University of Bonn, Germany, stefan.kollet@uni-bonn.de
\textsuperscript{2}Center for Applied Geosciences, Tübingen University, Germany, jensschum@googlemail.com
\textsuperscript{3}Center for Applied Geosciences, Tübingen University, Germany, claudius.buerger@uni-tuebingen.de
\textsuperscript{4}R\&H Environmental Ltd., Nürnberg, Germany, dboesel@rh-umwelt.de

Abstract

In cloud computing, software and data are shared via servers that can be accessed on-demand through basic terminals in conjunction with a Web browser. This affords the efficient utilization of software and hardware infrastructure by multiple users without the need of local software installation and maintenance. Here, this concept is applied and extended to the integrated hydrologic simulation platform ParFlow via a newly developed Web interface. The interface advances the concept of cloud computing by providing a comprehensive user interface, not only for the application of ParFlow, but also for its use in supercomputer environments without the direct involvement of the end-user. This also opens new possibilities for true grid computing, i.e. the simultaneous utilization of heterogeneous systems located at different geographic locations. The current version of the interface provides full functionality of ParFlow including the use of cluster resources. Input and output are handled via an intuitive GUI including comprehensive error catching. Simple visualization capabilities are provided through an interface with VisIt, an open source parallel rendering software that is also included in the service. In addition to ParFlow, optimization capabilities are available based on PEST and an EnKF data assimilation tool. Registered users can access the beta version of the service from anywhere in the world through a terminal in conjunction with a Web browser.

Introduction

There is no unifying definition of cloud computing in the literature to our knowledge. A quite generic definition provided by Wikipedia refers to cloud computing as the “provision of computational resources on demand via a computer network”. Here computational resources refer to both hardware and software resources that, in addition to being available on-demand as aforementioned, hide much of the technical details from the end user, such as Web services provided by e.g. Google and Amazon.

In science, cloud computing has been adopted quite rapidly. As a matter of fact, high-performance computing (HPC) i.e. the remote utilization of large-scale interconnected computer resources by a diverse user base, can be seen as a special case of cloud computing. However, applications in “The Cloud” are intuitively associated with on-demand computational resources and user interfaces as well as Web services that ensure a large degree of user-friendliness without limiting software and hardware capabilities. In science, user-friendliness often is of minor importance, but commences to play a more significant role in teaching and knowledge transfer from academia to the public and industry.

To date only limited literature on the potential and limitations of cloud computing in science is available. In recent articles, Sterling and Stark (2009) and Hunt et al. (2010) highlight the prerequisites that must be met for cloud computing becoming an integral part of science in connection with HPC, which are in part also relevant for the presented project. In the opinion of the authors, the main strengths of cloud computing stem from the utilization of remote resources without the disadvantage of initial financial investments and costs related to maintenance and upgrades of the systems; without difficulties in software installation and configuration of the systems; fast and on-demand access to functional, upgraded capabilities without time investment and debugging of the user base; cost savings through high-availability independent of geographic location; and a central environment for the management of very large data sets including post/pre-processing and visualization. On the other hand, important requirements of cloud computing in HPC environments are associated with the concept of virtualization and linked performance issues; parallel scalability; limitations in network communications; and security and dependability of the systems. An additional issue is the sustained allocation of the administrative resource for managing and maintaining the system and supporting the user base.

While the rational of this project is based on some of the thoughts presented above, it is not the objective to address all the pertinent issues related to cloud computing in scientific applications. The primary goal is a first attempt to development an intuitive Web interface for the integrated hydrologic simulation platform ParFlow (Jones and Woodward, 2001; Kollet and Maxwell, 2006) in HPC.
environments that allows the user to access scientific software and hardware resources on-demand from anywhere in world via a terminal in connection with a Web browser. In the following, the development strategy and implementation of various capabilities are discussed supported by selected screenshots. So far applications at the beta level deal with simple simulation and optimization problems including post- and pre-processing (visualization) to demonstrate the usefulness of the system.

**Basic Development Strategy, Implementation, and Description of System’s Components**

In the development of the Parflow Web service and interface, the following specifications were defined as guidelines to arrive at a functional beta version for demonstration and application purposes:
- no involvement of the end user with regard to software, hardware configurations and management of HPC resources
- full functionality of ParFlow including parallel simulations
- I/O handling via an intuitive graphical user interface with data up/downloading
- basic optimization capabilities
- basic visualization of output

**Basic setup**
The system consists of a standard Linux cluster where ParFlow has been installed and tested following standard procedures. A server hosting the ParFlow Web service and interface is connected to the cluster and the Internet constituting the gateway for registered users of the system. The Web service communicates with the cluster via a collection of scripts that submit and control simulation jobs using SLURM (https://computing.llnl.gov/linux/slurm), an open-source derivative of the Parallel Batch System, PBS, for cluster resources management. The simulation jobs are defined and submitted by the user using the Web interface which is based on the Model-Viewer-Controller (MVC) architectural pattern and is explained in more detail below. At this point the cluster and the upstream server provide all the requirements for setting up hydrologic models, performing simulations, storing output and also visualization. It is clear that the current hardware resources are only able to handle a small user base. However, the use of e.g., scaleable HPC utilities such as SLURM ensures that the system is extendable without major changes in the communication infrastructure of the Web service.

**The Graphical User Interface**
The MVC architectural pattern was implemented by using Ruby on Rails as the Web framework. Additionally, JavaScript was utilized to enhance the user interface. On the server, the logical steps of model generation and cluster communication were realized. Communication between Web server and browser was realized with a REST-style architecture (Fielding 2010). By implementing a REST-style architecture, it was also ensured that all the cluster functionalities are exposed as an application programming interface (API) to any third party software. This API can be accessed from any device connected to the internet since it is based on the HTTP (Hypertext Transfer Protocol). The webserver to cluster communication was handled with ssh system calls where SLURM commands are executed in order to manage numerical simulations on the cluster. This information is provided in near real-time to the user. The view component of the MVC architectural pattern was implemented as a single web page application where all communication between webserver and web-browser is handled via AJAX (asynchronous JavaScript and XML; http://ajaxpatterns.org) calls.

The webclient was programmed using the Javascript framework JQuery (http://jquery.com/) and the backbone.js library (http://documentcloud.github.com/backbone/). All server objects are imaged onto the client and locally represented as objects and collection of objects. Synchronization is done when new user input is provided and saved. Here, the objects maintain a list of observers that act on any changes of the status of the model or collection of models. Save/update and delete calls are received via the REST interface by the servers and executed. Views are required to show a list of the model objects and to modify individual objects. Modifications are done with HTML templates which are generated or transferred by the server using AJAX calls.

**Work flow: model generation and simulation**
Upon registration and login, three model examples are provided to the user via “Models” tab (Figure 1). These models can be used as a starting point for customization or the user can create a new model from scratch by clicking the “New model” button. A model is activated with a click in the “Name” column of the Model table. This will show a short description of the model and make available the
input database including e.g., the model geometry, input parameters, timing information, pumping well information and solver variables.

Figure 1. The “Models” tab under which all existing models of a user are listed including a short description, the output from the last simulation and the current status.

Figure 2. Screenshot of the “Edit Model – Geometry Inputs” tab which allows the user to define different model geometries that can be used to represent e.g., topography and aquifer heterogeneity.
The data base can be edited via the “Edit Model” tab that leads to various additional tabs under which the aforementioned model attributes can be modified (Figure 2). Under the “Processes” tab, the user can also specify the number or processors that will be used in the simulation in the HPC environment. Required information simply consists of the number processors in the x, y and z-directions that implicitly define the spatial decomposition of the model for parallel computations. Note, the tabs follow the ParFlow naming convention i.e. each tab name can be found in the ParFlow manual to guide the user in the generation of a model. Upon completion, the model can be submitted to the cluster via the “Run” tab, which also includes near real-time logging of the status of the simulation during runtime (Figure 3). Additional options that can be connected to a model are optimization and data assimilation tools that are described in more detail below. After completion of a simulation the results can be downloaded or visualized using the basic post-processing tools which are also outlined below.

**Optimization Capabilities**

The optimization capabilities include the non-linear regression algorithm PEST (DOHERTY, 2004) and a data assimilation interface for ensemble Kalman filtering, EnKF, based on the algorithm by Evenson (2003). Both optimization options are accessible through “Optimization” and “Data Assimilation” tabs of the Web interface.

In case of the optimization using PEST, non-linear regression is performed on steady-state hydraulic head observations that need to be defined for individual observation locations. Here, calibration is performed on the location and withdrawal rates of pumping wells instead of hydraulic parameters commonly used in classical calibration problems in hydrogeology. The initial guesses for the well locations and pumping rates are also provided by the user under the same tab.

The EnKF interface allows to assimilate point measurements of soil moisture in time, which are uploaded via the Web interface. The algorithm requires a number of input parameters such as the assimilation time steps, ensemble size, and model variance. Time management and application of the filter is done by a Fortran program which is also installed on the cluster. The ensemble is created by generating equally likely realizations of subsurface heterogeneity using the turning bands algorithm implemented in ParFlow with ensuing simulations. The simulation results are then used by the filter to estimate the background error covariance matrix and correct the model state i.e. the 3D soil moisture distribution.
Visualization

Basic visualization was implemented using the parallel software VisIt (https://wci.llnl.gov/codes/visit/) for efficient visualization of very large data sets. VisIt has been developed at the Lawrence Livermore National Laboratory and is an open-source software designed for parallel rendering of structured and unstructured grids with additional data analysis capabilities. VisIt has been installed on the cluster and communicates with the server via password-free ssh.

Under the “Postprocess Model” tab, the user has the option to define different variables and hydraulic parameters for plotting. The basic plot types available at this point are 3D pseudo color volume and contour plots including cross sections with adjustable thresholds. Contour intervals can be specified by the user at constant and varying intervals. Sliders are available to choose individual time slices and cross sections in the x, y, and z-directions. After specification of the plot attributes, the user submits the job for plotting via the submit button. The generated plot is returned for ad hoc viewing in the same browser window. Figure 4 shows a screen shot of the “Postprocess Model” tab with a simple contour plot of hydraulic head of an ambient flow field influenced by mild aquifer heterogeneity.

![Figure 4. The “Postprocess Model” tab showing the basic options for visualization (left panel). Visualization example of a contour plot of hydraulic heads resulting from pumping in an unconfined aquifer with zonal heterogeneity (right panel).](image)

Conclusions

The project demonstrates that the framework of cloud computing is useful in providing scientific software in connection with high-performance computing resources to a broad user base. The integrated hydrologic simulation platform ParFlow has been implemented in this framework including an intuitive graphical user interface, and optimization and basic visualization utilities. Registered users can access the system via a terminal and Web browser independent of their geographic location. In the future, additional tools will be made available to simplify the generation of complex model geometries and to perform advanced scientific visualization.

References


