Jun 18th, 9:00 AM - 10:20 AM

# Back-end Science Model Integration for Ecological Risk Assessment

Tao Hong
*Oak Ridge Institute for Science and Education*, hong.tao@epa.gov

Chancellor Pascale
*Johns Hopkins University Applied Physics Laboratory*, chancellor.pascale@jhuapl.edu

Jonathan Flaishans
*Oak Ridge Institute for Science and Education*, flaishans.jonathan@epa.gov

Marcia Snyder
*Oak Ridge Institute for Science and Education*, snyder.marcia@epa.gov

S. Thomas Purucker
*Johns Hopkins University Applied Physics Laboratory*, puruker.tom@epa.gov

*International Environmental Modelling and Software Society (iEMSs)*
*7th Intl. Congress on Env. Modelling and Software, San Diego, CA, USA,*
*Daniel P. Ames, Nigel W.T. Quinn and Andrea E. Rizzoli (Eds.)*
*http://www.iemss.org/society/index.php/iemss-2014-proceedings*

# Back-end Science Model Integration for Ecological Risk Assessment

Tao Hong[1], Chancellor Pascale[2], Jonathan Flaishans[1], Marcia Snyder[1], S. Thomas Purucker[3]

*[1]Oak Ridge Institute for Science and Education Hong.Tao@epa.gov, Flaishans.Jonathan@epa.gov, Snyder.Marcia@epa.gov*
*[2]Johns Hopkins University Applied Physics Laboratory Chancellor.Pascale@jhuapl.edu*
*[3]U.S. EPA Purucker.Tom@epa.gov*

Abstract: The U.S. Environmental Protection Agency (USEPA) relies on a number of ecological risk assessment models that have been developed over 30-plus years of regulating pesticide exposure and risks under Federal Insecticide, Fungicide, and Rodenticide Act (FIFRA) and the Endangered Species Act. Since computing technology have changed dramatically over this time period, constituent legacy models often contain algorithms based on source code with defunct dependencies and/or have been integrated with graphical user interface elements no longer compatible with current operating systems. Model migration to modern web applications creates integration challenges for back-end science model code residing on a server. An example of a legacy FORTRAN code that we integrated into our regulatory web application (the übertool) is the Pesticide Root Zone Model (PRZM). PRZM is comprised of three components: a cloud-based server to host a browser based graphic user interface (GUI) as a front-end, a database to store model results, and a REST API which invokes the model. A significant advantage of cloud-based PRZM is that hardware operating system compatibility issues (an important obstacle in running legacy FORTRAN models) are avoided because simulations are executed on the cloud server. In addition, migrating to the cloud reduces development and maintenance costs, facilities model coupling and offers instant deployment of updates and new versions. With the database element, users can save simulation outputs or retrieve previous assessments for comparison.

***Keywords**: Cloud computing, Web application, Ecological risk*

## 1 Introduction

The USEPA relies on ecological risk assessment models developed over 30-plus years of regulating pesticide exposure and risks under Federal Insecticide, Fungicide, and Rodenticide Act (FIFRA) and the Endangered Species Act. Typically, a complete ecological risk assessment for a pesticide involves evaluating multiple models from different, often incompatible, software implementations. In addition, some constituent legacy models contain algorithms with defunct source code dependencies and graphical user interfaces that are difficult to integrate with current operating systems. In this circumstance, a risk assessor may have to conduct simulations on multiple versions of Windows operation systems in addition to managing the difficulties in coordinating multiple input format requirements. Compatibility and interoperability thus becomes an unnecessary technical hurdle for finding trans-disciplinary solutions to complex environmental problems (Laniak et al. 2013).

A modern solution is to integrate models with cloud computing technology (Yu et al. 2013; Morris et al. 2014; Walker et al. 2014). The well-documented time and cost efficiencies are driving adoption of cloud computing applications across both business and government enterprise systems (Council C.I.O. 2012). In addition, and of considerable interest for scientific and regulatory applications, is the ability to deploy computational tools in an agile manner that can scale rapidly and in ways that current scientific desktop applications cannot. Based on the definition of National Institute of Standards and Technology (NIST), cloud computing is “a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" (Mell et al. 2011). Determined by level of services provided, cloud computing has three categories: Software-as-a-Service (SaaS), Platform-as-a-service (PaaS), and Infrastructure-as-a-service (IaaS). SaaS hosts pre-configured applications which are directly accessible to end-users (e.g., webmail). PaaS provides a platform, on which users can develop, test and deploy applications. However, users do not have access to the lower infrastructure level. IaaS "rents" hardware resources (e.g., CPU, storage, etc.) to customers. On any given IaaS cloud, users can install software without restrictions, but they must manage operating systems, data storage, firewalls, etc. Comparisons between SaaS, PaaS, and IaaS are shown in Table 1 (Rackspace Support 2013; Apprenda 2014). After evaluating features and restrictions across different cloud computing service models, we decided to develop a SaaS product whose front-end and back-end are based on PaaS and IaaS platforms, respectively. Hosting both ends separately increases flexibility and reliability of the system. Under this architecture, any changes or even system migration happened on one end are independent, which will not jeopardize the normal service provided by the other end.

Table 1. Comparison of different Cloud Computing Service Models

| Type | Feature | Not suitable | Example |
|---|---|---|---|
| Software-as-a-Service (SaaS) | • Directly delivers applications to end-users<br>• Users not required to handle software upgrades<br>• Easy accessibility (e.g., through web browsers) | • May need to be internally hosted due to regulation and/or security concerns | • Microsoft Office 365<br>• Webmail<br>• Google Docs |
| Platform-as-a-service (PaaS) | • Developers can test and deploy applications<br>• Stable and scalable | • Application needs to be highly portable<br>• Application needs to be coded in certain programming languages | • Google App Engine<br>• Amazon Elastic Beanstalk<br>• Microsoft Azure Cloud Services<br>• OpenShift |
| Infrastructure-as-a-Service (IaaS) | • Users have full control over the infrastructure<br>• Users can scale the system dynamically<br>• Variable cost, utility pricing model | • Higher server management burden on developers | • Amazon Elastic Compute Cloud<br>• Google Compute Engine<br>• Microsoft Azure |

The front-end of the übertool provides user access to all the constituent models (Flaishans et al. 2014). Thirteen non-FORTRAN based models have been re-coded in Python and are also deployed on the front-end. The back-end is based on an IaaS platform designed to host databases (e.g., MongoDB) and seven legacy FORTRAN models that are more computationally demanding. The end product is a cloud-based tool, the übertool (www.ubertool.org), which includes a number of heavily-used ecological exposure models for pesticide risk assessments (Purucker et al. 2014). Übertool avoids hardware and operating system compatibility issues because it is browser-based and all simulations happen occur on cloud servers. Going forward, leveraging the cloud in the manner reduces the development cost for incorporating new models and the overall maintenance burden. From a regulatory perspective, it also offers standardized output, instant deployment of updates, and quick incorporation of new models which can result in higher quality assessments. A schematic of the overall architecture for the übertool cloud application is shown in Figure 1.

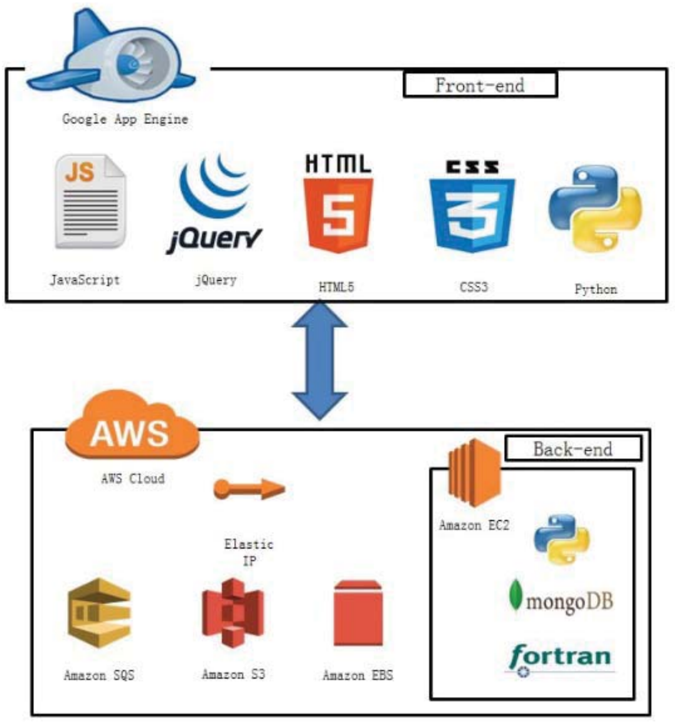


Figure 1. Schematic diagram of the architecture of the prototype übertool

## 2 Back-end integration

Two types of back-end integration are considered for übertool at the level of system infrastructure and models. The system infrastructure integration mainly focuses on creating a stable production environment. This environment would ensure that information is successfully transferred and shared within different back-end components as well as communication on the front-end. This process includes selection of a cloud platform, configuration of servers and databases, and installation of production environment, middleware, and applications. The second type of integration concentrates on risk assessment models, with a goal to host all of them as a service. This allows all the models to be callable from übertool front-end, but also accessible via an authentication process for other developers' products from a different web user interface or a desktop application. Hosting models as a service has the additional benefit of increasing interoperability with other environmental models and data sets.

The back-end infrastructure integration starts from hosting übertool on a cloud-based server. The advantages are it offers flexibility similar to a local server and takes care of hardware maintenance issues for developers; is highly reliable and instantly scalable based on demands; reduces up-front infrastructure expenses; and provides multiple valuable Application services such like long-term storage, task management, etc. (Berriman et al. 2013; Varia et al. 2014). The structure of the back-end server is shown in Figure 2, where three types of Amazon Web Services (AWS) products are integrated. The primary component is Amazon Elastic Compute Cloud (EC2) since it: hosts the platform for data transfer between the front and back-end through Representational state transfer (REST) application programming interface (API); creates the environment for Fortran models to be executed; generates reports and documents; hosts a NoSQL database (MongoDB) to manage user history; and runs a Python-based task queue management system (Celery). Other elements on AWS include Simple Storage Service (S3) and Simple Queue Service (SQS), where the former is designed to store model output files for the long term and the latter manages requests generated from batch model runs.

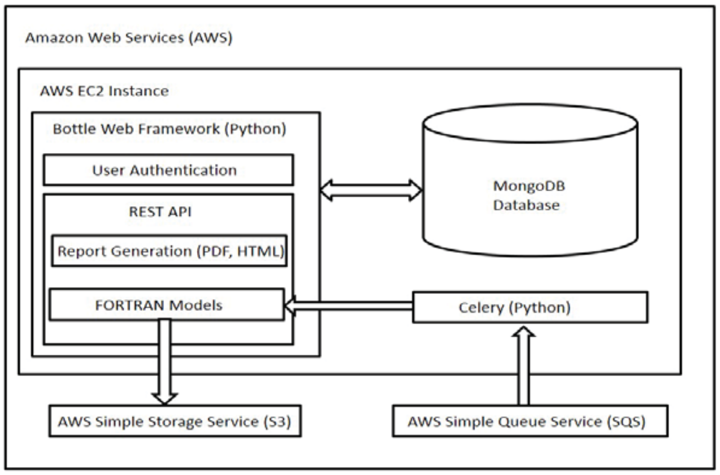

Amazon Web Services (AWS)
AWS EC2 Instance
Bottle Web Framework (Python)
User Authentication
REST API
Report Generation (PDF, HTML)
FORTRAN Models
MongoDB Database
Celery (Python)
AWS Simple Storage Service (S3)
AWS Simple Queue Service (SQS)

Figure 2. Structure of the back-end server

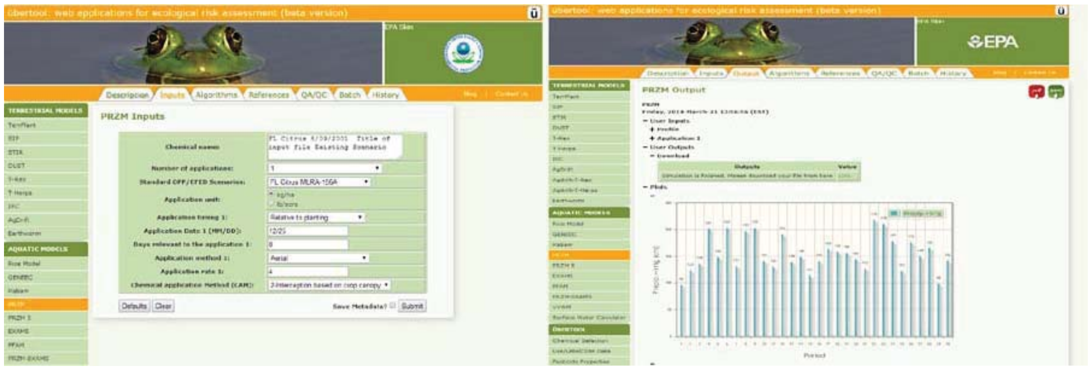
The REST API for übertool is built on Bottle, a lightweight Web Server Gateway Interface (WSGI)-based Python framework (Hellkamp 2014). Bottle contains a single file and has no dependencies other than the Python Standard Library. While some full-stack web frameworks (e.g., Django and web2py) support a number of activities through build-in modules (e.g., interpreting requests, producing responses, storing data persistently, etc.), Bottle could achieve these features through plug-ins which enables developers to customize web applications easily. Conceptually, the REST API performs as a “hub” on the back-end server which receives requests and inputs from the front-end, distributes requests by invoking different functions, and transfers outputs to the front-end for display. Access to the REST API is controlled by a user authentication system to prevent unauthorized users from changing or deleting data on the server. All data transmitted over the REST API follow the JavaScript Object Notation (JSON) format commonly used in modern web applications (Crockford 2009). Three types of APIs are created: 1) executing legacy FORTRAN models; 2) generating reports based on model outputs (PDF and HTML format); and 3) exchanging information between a non-relational database, MongoDB. A noSQL database was preferred over a relational database because it is convenient to modify the existing data structure (e.g., introducing new fields, changing structure, etc.); has fast query speed, and is easy to scale horizontally through automatic sharing and building replications (Amol 2014)

The model level back-end integration is fulfilled by APIs. The developed APIs not only facilitate data flow between front-end GUI and back-end services, but also increase data sharing with other sites and applications, since they "expose" some of a program's internal functions to the outside world. This idea is similar to the Google Map API which is used by numerous sites to show directions. Inside übertool, API methods “wrapping” the FORTRAN models can be integrated by other platforms, for example other ecological risk assessment web applications or stand-alone software running on the desktop (e.g., R, Python). Sharing FORTRAN models via APIs reduces development costs since all one needs to pay attention to is the data input/output structure and front end construction. In addition, the accuracy of calculations based on EPA released APIs can be verified via a continuous integration process. In the future, it could be a trend for regulatory agencies to both host scientific models on a cloud-based platform for the general purpose and also release these models’ API to meet users’ specific requirements.

## 3 Example

An example of the back-end service integration is demonstrated through a legacy, but heavily used FORTRAN model, Pesticide Root Zone Model (PRZM). PRZM is a one-dimensional, dynamic, compartment model capable of predicting the fate and transport of pesticides in unsaturated soil systems around the plant root zone, and its major components are hydrology and chemical transport. The hydrologic component estimates runoff, erosion, evaportranspiration and water movement, while the chemical component distributes pesticides, and organic as well as inorganic species, in the soil (Carsel et al. 1985; Carbone 2002; Suárez 2005).

Screenshots of the PRZM graphic user interface (GUI) appear in Figure 3. Some of the dynamic effects are realized through JavaScript and jQuery libraries. For example, the 'jQuery Validation' validation library (Zaefferer 2014) checks the quality of user inputs to prevent invalid entries. The jQuery library 'jqPlot' (Leonello 2013) creates graphics for visualizing model output. The layout of the model page contains three columns. The left panel is the navigation section with links to a collection of übertool models. Thus, the first step in conducting an ecologic risk assessment on the übertool is to choose the target model. The central panel frames of the webpage is designed to display model related information from eight perspectives with tab names of: 1) description, which introduces model objectives and limitations; 2) inputs (Figure 3, left), which accepts user inputs and pass them to models to conduct a one-time simulation (single run); 3) outputs (Figure 3, right), which only appears when a model simulation is complete and displays results as tables, plots and links for output file download; 4) algorithms, which contains all the governing equations; 5) references, which compiles resources on model manuscripts, source codes, and relevant journal publications; 6) quality assurance and quality control (QA/QC), which verifies the quality of developed models by comparing model outputs against multiple test cases; 7) batch, which facilitates users to evaluate the model under a number of scenarios and a comma-separated values (CSV) input template is required; and 8) history, which summarizes users' previous model simulations as records in a table. The history (output) page can be retrieved by clicking on the record. The right panel serves as the information and education zone, which collects a number of links related to ecological risk assessment.



| | |
|---|---|
| Chemical name: | FL Citrus 6/30/2001 Title of input file Existing Scenario |
| Number of applications: | 1 |
| Standard OPP/EFED Scenarios: | FL Citrus MLRA-156A |
| Application unit: | kg/ha |
| Application timing 1: | Relative to planting |
| Application Date 1 (MM/DD): | 12/25 |
| Days relevant to the application 1: | 0 |
| Application method 1: | Aerial |
| Application rate 1: | 4 |
| Chemical application Method (CAM): | 2-Interception based on crop canopy |

Figure 3. Screenshots of PRZM graphic user interface
(left. input page; right. output page)

All the client-side communication in übertool is illustrated in Figure 4 using PRZM as an example. When a user visits the PRZM input page through a browser, a GET request is sent to the front-end server and the server will respond with HTML, CSS, jQuery library files and default parameters to display the input page. To conduct a single model run, a user provides input parameters and submits them through a POST request to the server. Once the server receives the model inputs, it invokes the PRZM model simulation by passing parameters to the REST API on a server dedicated to hosting the simulation models. When the calculation is finished, model output files are transferred to a cloud directory for long-term preservation, while important output values and URL from the cloud repository will be transferred back to the user's browser along with HTML, CSS, and jQuery libraries for output display. The last step for this single model simulation is to save the PRZM object (inputs and outputs), output page html, and the cloud storage URL to the noSQL database. This is done through another REST API call from the front end to the model server. The REST API call is always initiated through the back-end servers instead of a client browser is because the latter option is less secure and could expose the credentials of the REST server.

The batch run mode is used when a user needs to run multiple PRZM simulations. In this case, all the inputs across different scenarios are coded in a formatted CSV template. After uploading this template from the browser, each row is parsed and validated by the front end server . Each row is considered as a single run task, and is placed in a task queue. At the same time, a Python-based task consumer, Celery, monitors the status of this task queue and assigns every task to the PRZM REST

API, which invokes the PRZM simulation as resources become available. Essentially, a batch model run is treated as a collection of single model runs, but only one batch object (no HTML, CSS, etc.) will ultimately be stored to the database. The benefit of introducing the task management system is to reduce inbound traffic at peak times, but users may have to check the batch run output later from the history page for longer simulations.

The history page allows a user to revisit previous submitted model runs. Upon loading this page, a query is initiated with a user ID and a model name (PRZM). All the qualified records will be returned from the database and transferred to the front end where a summary table is generated. By clicking different records, historical output page will be retrieved or regenerated based on stored object. To retrieve output page for a 'single' or 'qa/qc' simulation, a query is performed with specific job ID, which was generated on the submission of the simulation. The database would respond with previously stored HTML, CSS, and jQuery and push them to the browser for display. If a 'batch' simulation is queried, the response will be output objects and the output page will be 'assembled' in the browser for user review.

In addition to PRZM, similar back-end service integrations have been applied to seven other FORTRAN models. These models include GENEEC (Parker et al. 1995), EXAMS (Burns et al. 1981), PRZM-EXAMS (Burns et al. 1981; Carsel et al. 1985), PFAM, PRZM5, VVWM, and Surface Water Calculator. Creating APIs for those legacy models over the cloud facilitates access, management, and integrated with other data and model web services. For example, a compound FORTAN model, PRZM-EXAMS, is included in übertool, which is designed to estimate pesticide concentrations in surface waters for drinking water and aquatic exposure assessments. When this compound model is executed, the PRZM API is invoked first, followed by EXAMS, since PRZM outputs serve as inputs to EXAMS. Coupling different models through APIs is very convenient, since a developer reuses previously developed code and is primarily concerned with managing input/output. This can be different from a traditional model integration effort, where one needs to invest more effort in the integration of software's infrastructure.

## 4 Discussion and Conclusion

Although hosting scientific applications over the cloud is becoming increasingly popular, not every application is suitable for this migration. The following aspects should be evaluated before deciding: 1) Platform Selection, typically is a choice between PaaS and IaaS. PaaS may be preferred if the application to be hosted is not computationally demanding and its underlying development environment is suitable for the PaaS platform (programming language, data storage, etc.) being considered. Applications deployed on a PaaS platform would gain stability and scalability, since experienced service providers will manage and optimize the infrastructure for developers. However, if the application is more computationally intensive or composed of legacy code which requires customization, IaaS may be a better choice. High performance computing centers are still a favourable solution for high computational intensive applications. Because it offers machines and storages for almost any purposes (Jackson et al. 2011; Truong et al. 2011). 2) Data Confidentiality, if sensitive data is processed or generated by the application, it may not appropriate to host this application on a public cloud. Because sometimes cloud data could be stored, cached and transferred among different nodes to maximize cloud's efficiency (Fox et al. 2009; Truong et al. 2011). Hybrid clouds may be created to handle additional security needs. 3) Cost, it is complex to compare the overall cost of using cloud service against in house (on-premise) hosting, because some costs/benefits components with cloud hosting are hard to estimate and cloud providers are continually introducing lower pricing models. In general, the economic advantages of moving to the cloud includes: lower initial infrastructure and resource costs, less management responsibility since system providers will take over some management concerns (e.g., system purchase, maintenance, upgrade, etc.), reduced operational expenditure, cloud users are exempted from real-estate cost of data centers, electricital utility bills, salaries for system administers (IaaS platform does require a certain level of system administration knowledge) (Tak et al. 2011; Truong et al. 2011; Jula et al. 2014). In general, migrating applications to the cloud will attract more and more attention in the future, but is not a simple task, and in certain situations may not be more economical than tradition in-house hosting.

The übertool is one of the earliest cloud-based scientific applications, and is built to function as a "one-stop shop" by EPA regulatory program offices completing assessments. In addition, it is accessible by federal, industry, and academic researchers outside the agency. It combines disparate software models into an integrated web application and hosts them on a coherent platform so the models can be more easily parameterized, efficiently run, and transparently documented. The availability of the übertool may also facilitate the adoption of EPA models by other national governments and therefore contribute to "harmonization" -- consistent international regulation of chemicals -- especially within NAFTA, South America, and Europe. The übertool also has education properties, and it can be useful in traditional lectures since some of the algorithms' prototypes are adapted from textbooks (Hong et al. 2012).

## 5 Acknowledgments

This research was supported in part by an appointment to the Postdoctoral Research Program at the Ecosystems Research Division administered by the Oak Ridge Institute for Science and Education through Interagency Agreement No. (DW8992298301) between the U.S. Department of Energy and the U.S. Environmental Protection Agency.

## 5 REFERENCES

Amol. (2014). "Web Frameworks for Python." Retrieved March 14th, 2014, from https://wiki.python.org/moin/WebFrameworks.

Apprenda. (2014). "IaaS, PaaS, SaaS (Explained and Compared)." from http://apprenda.com/library/paas/iaas-paas-saas-explained-compared/.

Berriman, G. B., E. Deelman, et al. (2013). "The application of cloud computing to scientific workflows: a study of cost and performance." Philosophical Transactions of the Royal Society a-Mathematical Physical and Engineering Sciences **371**(1983).

Burns, L., S. Cline, et al. (1981). "Exposure Analysis Modeling System (EXAMS): User Manual and System Documentation. US EPA." Environmental Research Laboratory, Athens, GA.

Carbone, J. P. (2002). "Predicting fate and transport: The pesticide root zone model." Environmental Toxicology and Chemistry **21**(8): 1533-1534.

Carsel, R. F., L. A. Mulkey, et al. (1985). "The pesticide root zone model (PRZM): A procedure for evaluating pesticide leaching threats to groundwater." Ecological Modelling **30**(1): 49-69.

Council C.I.O. (2012). "Digital government: Building a 21st century platform to better serve the American people." Retrieved March **15**: 2015.

Crockford, D. (2009). "Introducing json." Available: json. org.

Flaishans, J., T. Hong, et al. (2014). Fronting Integrated Scientific Web Applications: Design Features and Benefits for Regulatory Environments. International Environmental Modelling and Software Society (iEMSs) 7th International Congress on Environmental Modelling and Software, San Diego.

Fox, A., R. Griffith, et al. (2009). "Above the clouds: A Berkeley view of cloud computing." Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS **28**: 13.

Hellkamp, M. (2014). "Bottle." 0.12. from http://bottlepy.org/docs/dev/index.html.

Hong, T. and S. T. Purucker (2012). An integrated web-based assessment tool for assessing pesticide exposure and risks. Ecological Society of America 2012 Annual Meeting, Portland, Portland, OR.

Jackson, K., K. Muriki, et al. (2011). "Performance and cost analysis of the Supernova factory on the Amazon AWS cloud." Scientific programming **19**(2-3): 107-119.

Jula, A., E. Sundararajan, et al. (2014). "Cloud computing service composition: A systematic literature review." Expert Systems with Applications **41**(8): 3809-3824.

Laniak, G. F., G. Olchin, et al. (2013). "Integrated environmental modeling: A vision and roadmap for the future." Environmental Modelling & Software **39**(0): 3-23.

Leonello, C. (2013). "jqPlot." from http://www.jqplot.com/index.php.

Mell, P. and T. Grance (2011). The NIST Definition of Cloud Computing, National Institute of Standards and Technology.

Morris, D. E., J. E. Oakley, et al. (2014). "A web-based tool for eliciting probability distributions from experts." Environmental Modelling & Software **52**(0): 1-4.

Parker, R., H. Nelson, et al. (1995). GENEEC: a screening model for pesticide environmental exposure assessment. Proceedings of the International Exposure Symposium on Water Quality Modeling.

Purucker, S. T., T. Hong, et al. (2014). Decision Support and Web-based Implementation of Algorithms for the Ecological Assessment of Pesticides. International Environmental Modelling and Software Society (iEMSs) 7th International Congress on Environmental Modelling and Software, San Diego, CA, USA.

Rackspace Support. (2013). "Understanding the Cloud Computing Stack: SaaS, PaaS, IaaS." from http://www.rackspace.com/knowledge_center/whitepaper/understanding-the-cloud-computing-stack-saas-paas-iaas.

Suárez, L. (2005). "PRZM-3, a model for predicting pesticide and nitrogen fate in the crop root and unsaturated soil zones: users manual for release 3.12. 2." US Environmental Protection Agency (EPA), Washington, DC.

Tak, B. C., B. Urgaonkar, et al. (2011). "Understanding the Cost of Cloud: Cost Analysis of In-house vs. Cloud-based Hosting Options." The European Business Review: 76-80.

Truong, H.-L. and S. Dustdar (2011). "Cloud computing for small research groups in computational science and engineering: current status and outlook." Computing **91**(1): 75-91.

Varia, J. and S. Mathew (2014). Overview of Amazon Web Services

Walker, J. D. and S. C. Chapra (2014). "A client-side web application for interactive environmental simulation modeling." Environmental Modelling & Software **55**(0): 49-60.

Yu, B., R. Sen, et al. (2013). "An integrated framework for managing sensor data uncertainty using cloud computing." Information Systems **38**(8): 1252-1268.

Zaefferer, J. (2014). "jQuery Validation Plugin." from http://jqueryvalidation.org/.

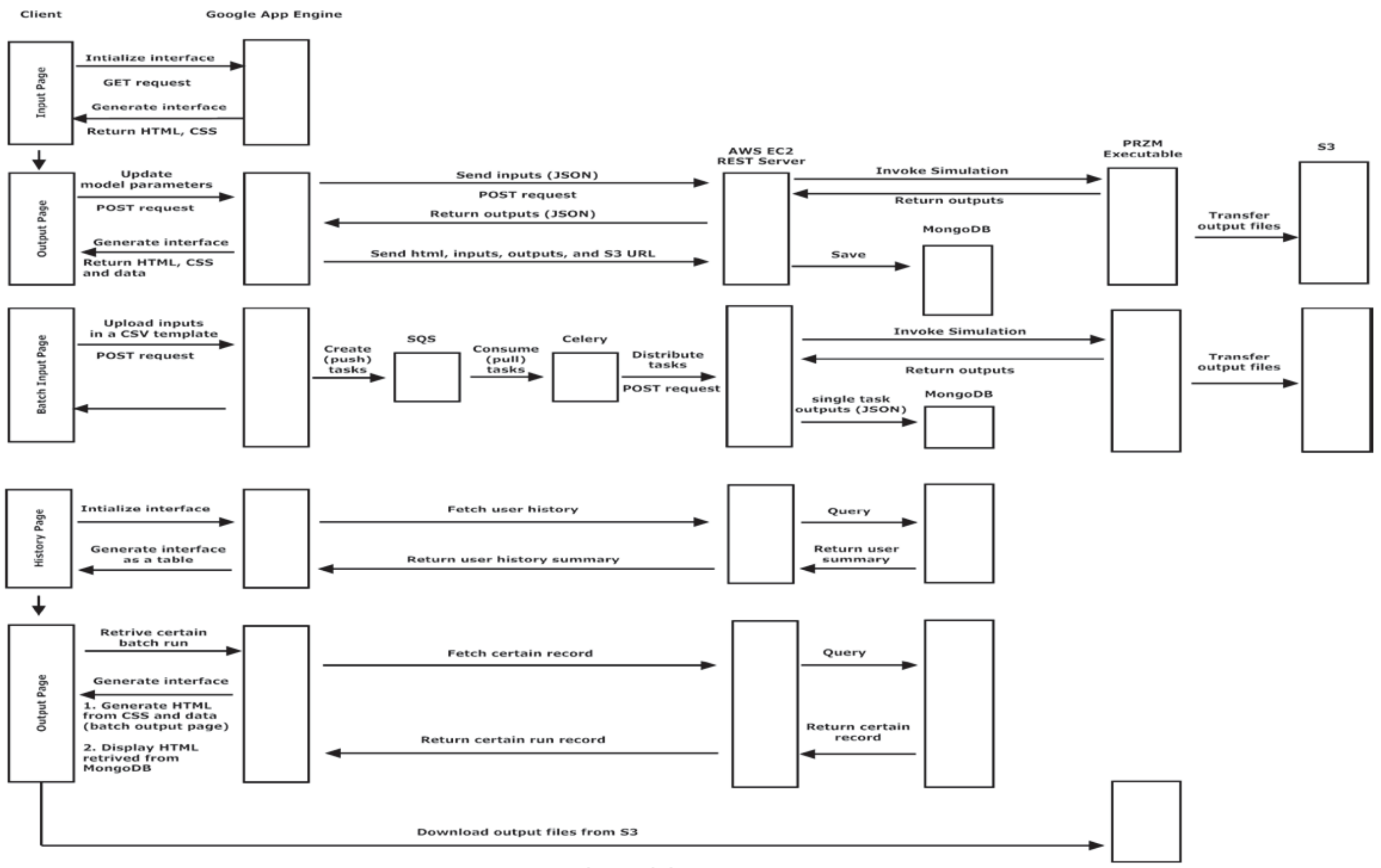


Figure 4. Workflow of Cloud-based PRZM Model