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

Model-as-a-Service (MaaS) using the Cloud Services Innovation Platform (CSIP)

O. David[ab], W. Lloyd[ab], K. Rojas[d], M. Arabi[a], F. Geter[d], J. Ascough[c], T. Green[c], G. Leavesley[a], J. Carlson[a]

[a] Dept. of Civil and Environmental Engineering, [b] Dept. of Computer Science, Colorado State University, Fort Collins CO, USA
{odavid, wlloyd, jack.carlson, ghleavesley, mazdak.arabi}@colostate.edu;
[c] USDA, Agricultural Research Service, Fort Collins CO, USA
{jim.ascough, tim.green}@ars.usda.gov
[d] USDA, Natural Resources Conservation Service, Fort Collins CO, USA
{ken.rojas, frank.geter}@ftc.usda.gov

Abstract: Cloud infrastructures for modelling activities such as data processing, performing environmental simulations, or conducting model calibrations/optimizations provide a cost effective alternative to traditional high performance computing approaches. Cloud-based modelling examples emerged into the more formal notion: "Model-as-a-Service" (MaaS). This paper presents the Cloud Services Innovation Platform (CSIP) as a software framework offering MaaS. It describes both the internal CSIP infrastructure and software architecture that manages cloud resources for typical modelling tasks, and the use of CSIP's "ModelServices API" for a modelling application. CSIP's architecture supports fast and resource aware auto-scaling of computational resources. An example model service is presented: the USDA hydrograph model EFH2 used in the desktop-based "engineering field tools" is deployed as a CSIP service. This and other MaaS CSIP examples benefit from the use of cloud resources to enable straightforward scalable model deployment into cloud environments.

Keywords*:* Modeling, Cloud computing, SOA, MaaS, REST

1 INTRODUCTION

The use of cloud computing for scientific computing and environmental modeling in particular gained substantial traction in recent years (Jha et al., 2012). A cloud is a large pool of available and easily accessible virtual resources such as hardware, platforms, and software services. Such resources can be allocated and disposed ad-hoc, their dynamic configuration to various requirements makes it attractive for scientists, research groups, organizations, and agencies to explore their potential for research projects and operational use.

Other appealing cloud features are (1) the absence of in-house maintenance and administration of such resources, (2) the availability of a range of competing vendors offering different pricing models, (3) the flexibility in adjusting applications or operating systems rapidly on a large scale. Other aspects such as (1) secure access and data protection, (2) governing the physical location of cloud-managed resources or (3) guaranteed availability had to be addressed by commercial cloud vendors to make it a viable option for operational use within the modeling community and not just academia.

Model-as-a-Service (MaaS) (Zou 2012) emerged as "a concept of being able to invoke re-usable, fine-grained software components across a network" (Roman et al., 2009). MaaS can be viewed as a "Software-as-a-Service" variant addressing the need to make (environmental) models available as a web-service. This approach enables a model "service provider" to lower the burden for model users

significantly by potentially provisioning model input data with the service and provide for an adequate execution environment.

The Cloud Services Innovation Platform (CSIP) project was established at Colorado State University (CSU) in collaboration with the USDA Natural Resources Conservation Service and Agricultural Research Service to explore the prospect of cloud computing for MaaS and related data management. The project is supported by USDA. The objective is to develop a scalable, modular, cost effective, and open deployment platform for simulation models while leveraging new and legacy research simulation models as cloud based web-services.

The remainder of the paper introduces the physical implementation of CSIP at CSU, the CSIP client API, and the server-side implementation of CSIP services. Finally, some modeling applications are presented.

2 MODEL-AS-A-SERVICE (MaaS)

A “Model-as-a-Service” provides the capability to execute simulation models as a service. MaaS solely focuses on the application aspect of a model against data. There are two main usage patterns: (i) The model can be pre-deployed, has a well-known service endpoint, and may be supported by supplemental data services. This is quite common for operational models used in a production environment. Moreover, (ii) the model can be dynamically deployed from the client before execution. Model service development for research purposes needs such a behavior. Both approaches address a different workflow, need for availability and security. A certain model execution method may also be specified in such a service.

MaaS involves web-services as the interface with which the client can communicate. Data is exchanged as structured text, such as XML or JSON (JavaScript Object Notation) in a specified syntax. However, data may also be passed to the service in the model’s native format.

3 CLOUD SERVICES INNOVATION PLATFORM (CSIP)

3.1 CSIP Cloud Approach

There are several approaches to cloud computing in general, usually categorized by the service capability and abstraction level that a cloud service provide. “Software as a Service” (SaaS), “Platform as a Service” (PaaS), and “Infrastructure as a Service” (IaaS) are those categories. IaaS represents the most flexible cloud approach for a user since access at the operating system level (networking, storage) is granted. For the implementation of a cloud-based modeling solution access to cloud supporting IaaS is considered important. For CSIP the integration of existing legacy models was critical. To make legacy models aware of a cloud environment is usually not feasible. Usually, models were not developed with cloud features in mind. As a result, only IaaS approaches support the migration into a cloud environment with respect to custom execution management, data source provisioning, etc. for legacy models, not vice versa.

Cloud providers such as Amazon, Google, or Microsoft offer cloud solutions that are publicly accessible and are therefore referred to as “public cloud”. A private cloud, however, is hosted within the organizational structure using internal IT resources. Private cloud software is available as open source and from commercial vendors, examples are Eucalyptus (Nurmi, 2009), OpenStack (OpenStack, 2012), or VMWare. Public and private clouds are similar from a technical perspective. A blend of private and public cloud use is considered a hybrid cloud, with the goal of seamless transition from private to public cloud use with increasing workload. Such an approach seems to fit well for modeling infrastructures offering such services. A baseline modeling workload can be handled with the private, in-house portion of the cloud. If the needed computational resources exceed the private capabilities, public cloud resources can be used to accommodate larger workloads. Such a solution is cost effective, since it only requires payment to service providers at peak times and leverages available in-house resources otherwise. A hybrid cloud is also the preferable approach during modeling service development, since service development cycles won’t tap into the service budget.

CSIP is setup as two physical private clouds, named “omscloud” and “eramscloud”, at Colorado State University using Eucalyptus (Figure 2). Eucalyptus is an open source “Infrastructure-as-a-Service” cloud. It is compatible with Amazon’s web-service APIs for EC2, S3, and others. CSIP is also using Amazon’s AWS as its public cloud part. CSIP virtual machine images are identical for the private and public cloud. The private cloud runs on blade server systems equipped with a total of about 150 compute cores, 1TB of memory and 24 TB of storage capacity.

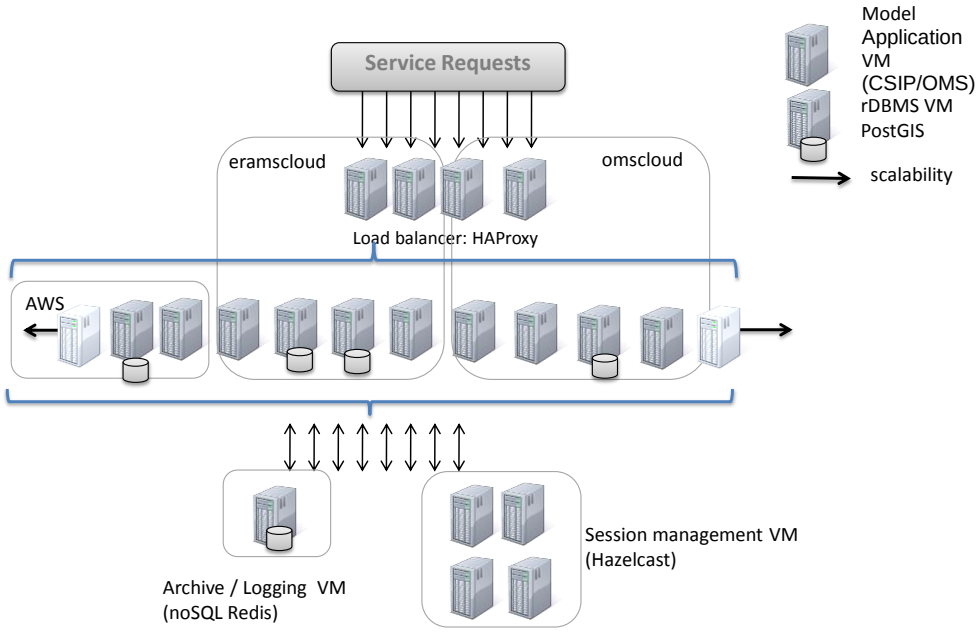

Figure 1. CSIP Physical Architecture.

Virtual images are configured with Tomcat[1] as container for the services, Haproxy[2] for service load distribution, the Redis[3] NoSQL database for archival of service sessions, and Hazelcast[4] for distributed session management. There is no single point of failure for model runs within CSIP because there is no central resource management. All components can scale individually and provide failover and redundancy in order to allow for maximum service availability.

The management of the CSIP cloud resources is performed by VMScaler, an open source cloud resource monitoring, management, and virtual node allocation/de-allocation software. This software is described in another paper at this conference (Lloyd et al., 2014).

3.2 CSIP MaaS Approach

CSIP provides a simple, open interface to its MaaS services. The CSIP APIs conform to the design principles of RESTful web services (Fielding, 2002). REST stands for Representational State Transfer and it is an architectural approach where software systems can be built in which clients can perform requests to service end points. REST is a widely used method to implement client/server architecture. It allows development of software applications that are simple to use, easy to scale, and highly inter-operable. CSIP Data is passed to and from the Web service as JSON data objects. Model parameter, execution results, meta-information, are all being passed as JSON objects from the client to the web-service. It provides support for catalog listing of models, exploration of model capabilities, and execution control. Such an approach ensures a maximum level of interoperability and flexibility between heterogeneous systems and applications.

The CSIP REST web service protocol resembles concepts of Web Processing Services (WPS) developed by the OpenGIS Consortium (OpenGIS, 2007). However, CSIP simplifies data definitions,

[1] http://tomcat.apache.org/
[2] http://haproxy.1wt.eu/
[3] http://redis.io/
[4] http://www.hazelcast.org/

data descriptions, and meta-data in order to allow an easy use of the offered web services. The CSIP web-service interface defines three operations that can be requested by a client and performed by a CSIP implementation.

- *Provide a list of available model and data services:* This operation allows a client to request and receive back service meta-data documents that describe the abilities of the specific implementation.
- *Describe a specific operation in detail:* This operation allows a client to request and receive back detailed information about the model/data service that can be run on the CSIP instance, such as inputs required, their allowable formats, and the outputs that can be produced.
- *Execute the model or data service:* This operation allows a client to run a specified process implemented by CSIP, using provided input parameter values and returning the outputs produced.

3.3 RESTful API

A CSIP model execution service expects model parameter as input in a JSON payload and optional metadata controlling the model execution. This service provides the model output result as a JSON object.

There are five execution lifecycle phases that are common for all CSIP model services. There are two main groups with respect to the actual model run and the management of model results. Since a model service can be submitted in two modes "sync" and "async" there are two variants of managing those phases. A model execution request submits a model run, which eventually transitions to a "Running" state. The model run can then either successfully terminate ("Finished"), it can fail ("Failed"), or be canceled by the user ("Cancelled"). The phases relate to output data management (if applicable). Model output data will be kept available for retrieval until it expires ("expiration_date"). The time to keep output data available can be controlled via the request metainfo entry "keep_results".

An example REST call that executes the EFH2 service is shown in Listing 1 below. The http header information shows the service end point, the target host and additional information for proper content negotiation for JSON. The result of this call is shown in part b) of Listing 2. The result contains according to REST requirements the original request parameter and an additional “result” section with the output values. The metainfo section now contains elements about the model run, such as a unique identificator “suid”, result expiration time (“expiration_date”), time stamp (“tstamp”), duration of the run (“cpu_time”), and status information about the success.

Listing 1. RESTful Model execution.

```
POST /csip-eft/m/efh2/1.1 HTTP/1.1
Host: <host>
Accept: application/json
Content-Type: application/json
```

A http POST execution in CSIP also supports the submission of input files to the request as multipart form data attachments. Model output files are being offered as URLs in the “result” section of the response and can easily be fetched using a http GET after the model is finished. Such file handing in CSIP simplifies and accelerates the integration of legacy models using their native formats.

Listing 2 shows the Request for a given service endpoint. Such a “template” can be obtained by issuing a http GET request against a service endpoint.

Listing 2: Model Service Example

a) Request

```
{
  "metainfo": {
     Keep_workspace: false,
     mode:"sync"
 },
  "parameter": [
    {
      "name": "precip",
      "description": "precip",
      "unit": "in",
      "min": "1",
      "max": "15",
      "value": 14
    },
    {
      "name":"runoffcurvenumber",
      "min": "40",
      "max": "95",
      "value": 90
    },
    {
      "name": "stormtype",
      "value": "I"
    },
    {
      "name": "watershedlength",
      "unit": "ft",
      "min": "200",
      "max": "26000",
      "value": 1500
    },
    {
      "name": "watershedslope",
      "unit": "%",
      "min": "0.1",
      "max": "64",
      "value": 0.5
    }
  ]
}
```

b) Response

```
{
 "metainfo": {
  "status": "Finished",
  "first_poll":1000,
  "next_poll":1000,
  "suid":"c6808036-db0b-11e3-84c6-8d184928a57a",
  "tstamp": "2014-02-14T02:02:17+0000",
  "service_url": "http:\/\/localhost:8080\/csip-
eft\/m\/efh2\/1.0",
  "cpu_time": 16,
  "expiration_date":"2014-05-14T02:07:17+0000"
 },
 "parameter": [
  {
   "name": "precip",
   "description": "precip",
   "unit": "inch",
   "min": "1",
   "max": "15",
   "value": 14
  },
  {
   "name": "runoffcurvenumber",
   "min": "40",
   "max": "95",
   "value": 90
  },
  {
   "name": "stormtype",
   "value": "I"
  },
  {
   "name": "watershedlength",
   "unit": "ft",
   "min": "200",
   "max": "26000",
   "value": 1500
  },
  {
   "name": "watershedslope",
   "unit": "%",
   "min": "0.1",
   "max": "64",
   "value": 0.5
  }
 ],
 "result": [
  {
   "name": "runoff",
   "value": 12.75
  },
  {
   "name": "timeofconcentration",
   "value": 0.727178
  },
  {
   "name": "unitpeakdischarge",
   "value": 0.3700218
  }
 ]
}
```

3.4 Model Service Implementation

Model services are implemented using the CSIP core library. It is developed using the Java Jersey RESTful framework, the JAX-RS reference implementation (Burke, 2014). A simple example of a service for the EFH2 model (USDA, 1987) is shown in Listing 3.

Listing 3. CSIP Service Implementation.

```
@Name("EFH2")
@Description("Storm runoff model based on conventions in Engineering Field Handbook.")
@Path("m/efh2/1.0")
@Polling(first = 1000, next = 1000)
public class V1_0 extends AbstractModelService {

  // the model
  EFH2HydrologyModel model = new EFH2HydrologyModel();

  @Override
  protected String process() throws Exception {
    Map<String, JSONObject> m = getParamMap();
    model.setPrecip(m.get("precip").getDouble(VALUE));
    model.setRunoffCurveNumber(m.get("runoffcurvenumber").getInt(VALUE));
    // … obtain more parameter here
    return model.simulate() == 0 ? EXEC_OK : EXEC_FAILED;
  }

  @Override
  protected JSONArray createResults() throws Exception {
    JSONArray result = new JSONArray();
    result.put(JSONUtils.data("runoff", model.getRunoffQ()));
    result.put(JSONUtils.data("timeofconcentration", model.getTimeOfConcentration()));
    result.put(JSONUtils.data("unitpeakdischarge", model.getUnitPeakDischarge()));
    return result;
  }
}
```

`AbstractModelService`, the superclass of every service handles session management, parameter parsing, file attachment extraction, temporary workspace creation and cleanup, error propagation, and threading of synchronous and asynchronous execution of the model. In addition to the `AbstractModelService` class there are other base classes that simplify the integration of any external executable or any Java class by requiring only the annotation of a service class with minimal coding. Such a “non-invasive” approach was derived for the Object Modeling System (OMS) component design and successfully adapted to the CSIP API (Lloyd et al., 2011).

4. APPLICATIONS

A multitude of CSIP services are implemented for exploratory use in research and operational application for USDA field offices and the public. CSIP services such as the Integrated Erosion Services that consist of the RUSLE2 (USDA ARS, 2008) and WEPS model (Hagen, 1991) are being called from Web applications such as (1) the Keystone Group Field to Market calculator, (2) eRAMS, a Web based GIS and modeling platform for integrated assessment of resources concerns, and (3) an extension for the ArcGIS based NRCS Customer Service Toolkit to obtain erosion estimates for use by 10,000 NRCS and conservation district field employees in daily operations. CSIP services are also being used to support research in watershed modeling with respect to model development, improvement and application in study areas. Specifically, the AgES-W watershed model is applied for watershed assessment in Colorado (Green et al., 2014) and Indiana (Ascough et al., 2014). AgES-W is a fully distributed watershed model developed and implemented using OMS as a modeling framework. AgES-W application in CSIP is carried out via the cloud enabled OMSConsole.

The Comprehensive Flow Analysis (CFA) toolbox (Wible and Arabi, 2013) is a comprehensive web-based solution for analyzing numerous aspects of streamflow. CFA’s analysis models supports examination of the various aspects of streamflow, including flood and drought frequency, duration curve analysis of streamflows, as well as base flow separation tools which isolate the influence of groundwater on stream discharge. All streamflow analysis methods are implemented as CSIP modeling services which are accessed from CFA’s user interface hosted within the eRAMS GIS platform.

Another integration of CSIP was funded by the International Atomic Energy Agency in 2013 and is currently in progress. A water balance model was enhanced with handling of isotopes, called

WBMIso, to eventually quantify the contribution of groundwater within the Nile basin. The model was developed under OMS version 3 and is being deployed as a service under CSIP. Other CSIP services currently in development include SWATDEG, a modified version of the Soil Water Assessment Tool (SWAT) to simulate widening on small alluvial and threshold streams, (ii) the deployment of the NRCS TR20 family of tools for storm event surface water hydrologic model at the watershed scale, (iii) the Landuse and Agricultural Management Practice Web-Service (LAMPS) (Kipka, 2013) that detects crop rotation sequences and matches them with available crop rotation information and associated tillage operation information to generate management input files for watershed models such as AgES-W over a simulation period.

Future anticipated applications of CSIP will focus on the integration of process-based agricultural system models (e.g., RZWQM) for climate change studies, services for nutrients and pesticides, and others. The paradigm shift from offering modeling software products to integrated modeling services is apparent.

5. CONCLUSIONS

The Cloud Services Innovation Platform (CSIP) provides Model-as-a-Service using a RESTful and JSON client interface protocols that support the most flexible integration into clients using any operating system, platform, and programming language. The CSIP protocol is straightforward to use as it provides a succinct standardized data transfer protocol for model input/output with metadata and optional legacy data formats.

CSIP has proven to be widely scalable and reliable due to its design. The lack of central management of web-services and its resources provided failover and redundancy features make it suitable to serve a large number of model executions performed by any number of users. Moreover, it serves modelling research needs by providing means for ensemble run management (streamflow prediction, uncertainty analysis, etc.). Recently, CSIP was implemented across hundreds of compute nodes for execution of climate change data sets. This use case showed no performance degradation and showed steady throughput. Many applications have been integrated as MaaS solutions using CSIP for modeling approaches facilitating the operational assessment of natural resource concerns and related research. Hydrology, erosion, energy, and water quality models have been integrated over the last couple of years. These efforts are ongoing.

CSIP was awarded the 2012 Computerworld Honors Winner for “Economic Development” (Computerworld, 2012) for its adeptness in research and technology transfer to facilitates the migration of existing scientific modeling applications into a cost-effective commercial and non-commercial solutions.

6. AVAILABILITY

The CSIP project including source code repositories for the core infrastructure as well as all service implementations (erosion, watershed analysis, water quality, flow analysis, etc.) are hosted at http://csip.javaforge.com. All source code is open source and can be downloaded from this site. CSIP is available under Lesser GNU Public License (LPGL). The site also provides a list of publicly accessible model services.

7. ACKNOWLEDGMENTS

CSIP research and development is being funded under a cooperative agreement 68-7482-13-518 between the USDA-Natural Resources Conservation Service and the Department of Civil and Environmental Engineering at Colorado State University.

8. REFERENCES

Ascough, J.C., Green, T.R., David, O., Kipka, H., McMaster, G.S., Fink, M., Krause, P., Kralisch, S., 2014. Advances in Distributed Watershed Modeling: A Review and Application of the

AgroEcoSystem-Watershed (AgES-W) Model, In: Daniel P. Ames, et al. (Ed.), 7th Intl. Congress on Env. Modelling and Software. International Environmental Modelling and Software Society (iEMSs): San Diego, CA, USA, this issue.

Burke, B., 2014, Restful Java with Jax-RS 2.0, O'Reilly Media, 2014.

Computerworld, 2012, “Computerworld Honors 2012 Winners”, http://www.computerworld.com/s/article/9227345/Computerworld_Honors_2012_Winners_Using_technology_to_benefit_society?pageNumber=3, 2012

David O., S.L. Markstrom, K.W. Rojas, L.R. Ahuja and I.W. Schneider, 2002, The Object Modeling System. In: L.R. Ahuja, L. Ma and T.A. Howell, Editors, Agricultural System Models in Field Research and Technology Transfer, Lewis Publishers, Boca Raton (2002), pp. 317–330.

Fielding, R.; Taylor, R., 2002, "Principled Design of the Modern Web Architecture", ACM Transactions on Internet Technology (TOIT) (New York: Association for Computing Machinery) 2 (2): 115–150.

GeoServices REST Specification Version 1.0, ESRI White Paper, 380 New York Street Redlands, California 92373-8100, 2010.

Grance, T., P. Mell, 2009, The NIST Definition of Cloud Computing, National Institute of Standards and Technology, vol. 53, no. 6, p. 50, 2009.

Green, T.R., Erskine, R.H., Ascough, J.C., Vandenberg, B., Pfennig, B. Kipka, H., David, O., Coleman, M.L., 2014. AgroEcoSystem-Watershed (AgES-W) Model Delineation and Scaling, In: Daniel P. Ames, et al. (Ed.), 7th Intl. Congress on Env. Modelling and Software. International Environmental Modelling and Software Society (iEMSs): San Diego, CA, USA, this issue.

Hagen, L.J., 1991, A wind erosion prediction system to meet user needs. J. Soil and Water Conservation. 46:106-111.

Jha, S, D. S. Katz, A. Luckow, A. Merzky, and K. Stamou, 2011, Understanding Scientific Applications for Cloud Environments, in Cloud Computing: Principles and Paradigms, R. Buyya, J. Broberg, and A. Goscinski, Eds. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2011, pp. 345-371.

Kipka, H., David, O., Lyon, J., Garcia, L.A., Green, T.R., Ascough II, J.C., and Rojas, K.W., 2013, A web-service tool to generate crop rotation management input files for spatially distributed agro-ecosystem models. In: Ramirez, J.A. (Ed.), Proc. Hydrology Days 2013, March 25-27, Fort Collins, Colorado. Colorado State University, Fort Collins, Colorado. p. 38-46.

Lloyd, W., David, O., Ascough II, J.C., Rojas, K.W., Carlson, J.R., Leavesley, G.H., Krause, P., Green, T.R., Ahuja, L.R., 2011, Environmental modeling framework invasiveness: Analysis and implications. Environmental Modelling & Software, 26(10), 1240-1250.

Lloyd W., O. David, M. Arabi, J. Ascough, T. Green, J. Carlson, K.Rojas, 2014, The Virtual Machine (VM) Scaler: An Infrastructure Manager Supporting Environmental Modeling on Infrastructure-as-a-Service Clouds, Proceedings of the 7th International Congress on Environmental Modelling and Software (iEMSs), 2014, this issue.

Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., Zagorodnov, D., 2009, "The Eucalyptus Open-Source Cloud-Computing System," Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pp.124-131, 18-21 May 2009.

Open Geospatial Consortium, Inc. OpenGIS Web Processing Service (WPS) Specification, 2007, WWW document, http://portal.opengeospatial.org/files/?artifact_id =24151, 2007.

OpenStack LLC, 2012, “OpenStack: The Open Source Cloud Operating System,” 21-Jul- 2012. [Online]. Available: http://www.openstack.org/software/.

Roman, D; Schade S.; Berre, A. J.; Bodsberg, N. R.; Langlois, J., 2009, Model as a Service (MaaS). In Grid Technologies for Geospatial Applications Workshop, Hannover, Germany, 2009. Available at http://ifgi.uni-muenster.de/0/agile/submissions/AGILE-GridWorkshop-Roman.pdf.

USDA ARS, 2008, Draft Science Documentation Revised Universal Soil Loss Equation Version 2. Washington, DC: USDA Agricultural Research Service. http://www.ars.usda.gov/sp2UserFiles/Place/64080510/RUSLE/RUSLE2_Science_Doc.pdf.

USDA Natural Resources Conservation Service (NRCS) Engineering Field Handbook (EFH), 1987, Chapter 2, Estimating Runoff and Peak Discharge

Wible T, M. Arabi, 2013, Comprehensive Flow Analysis Using Cloud-Based Cyberinfrastructure, Water Center of Colorado State University, (30):5, 2013.

Zou G., Zhang B., Zheng J., Li Y.; Ma J., 2012, MaaS: Model as a Service in Cloud Computing and Cyber-I Space, Computer and Information Technology (CIT), 2012 IEEE 12th International Conference on , vol., no., pp.1125,1130, 27-29 Oct. 2012.