# Real-Time Environmental Monitoring for Cloud-Based Hydrogeological Modeling with HydroGeoSphere

Lapin, Andrei ; Schiller, Eryk ; Kropf, Peter ; Schilling, Oliver ; Brunner, Philip ; Jamakovic-Kapic, Almerima ; Braun, Torsten ; Maffioletti, Sergio

DOI: https://doi.org/10.1109/HPCC.2014.154

# Real-time Environmental Monitoring for Cloud-based Hydrogeological Modeling with HydroGeoSphere

Andrei Lapin*, Eryk Schiller*, Peter Kropf*, Oliver Schilling†, Philip Brunner†,
Almerima Jamaković-Kapić‡, Torsten Braun‡, and Sergio Maffioletti§
*Computer Science Department (IIUN), Université de Neuchâtel, 2000 Neuchâtel, Switzerland
{name.surname}@unine.ch
†Centre for Hydrogeology and Geothermics (CHYN), Université de Neuchâtel, 2000 Neuchâtel, Switzerland
{name.surname}@unine.ch
‡Communication and Distributed Systems Group (CDS), Universität Bern, 3012 Bern, Switzerland
{jamakovi, braun}@iam.unibe.ch
§Grid Computing Competence Center (GC3), Universität Zürich, 8057 Zürich, Switzerland
sergio.maffioletti@gc3.uzh.ch

***Abstract*—This paper describes an architecture for real-time environmental modeling. It consists of a wireless mesh network equipped with sensors and a cloud-based infrastructure to perform real-time environmental simulations using a physics-based model combined with an Ensemble Kalman Filter. The purpose of the system is to optimize groundwater abstraction close to a river. These initial studies demonstrate that the cloud infrastructure can simultaneously compute a large number of simulations, thus allowing for the implementation of Ensemble Kalman Filters in real-time.**

## I. Introduction

Environmental systems are characterized by a significant degree of spatial heterogeneity, and are typically driven by highly dynamic boundary conditions. Quantifying and predicting the interactions between surface water and groundwater, for example, requires accurate information on the water table in the aquifer, the surface water dynamics, the physical properties of the stream bed and the underlying aquifer, as well as a numerical approach to simulate the fluid dynamics. Systems that allow us to quantify such interactions therefore must be able to i) capture data, ii) simulate the system state, and iii) provide predictions on the behavior of the ecosystem. Field sites of high interest are sometimes situated in remote locations. Therefore, having a real-time measurement infrastructure in place that is connected to the Internet and provides remote access to recorded data is highly beneficial [1], [2]. Models that are coupled to real-time measurement infrastructure are well suited for the prediction of hydrological systems. A range of numerical models that simulate surface water dynamics exist [3], and have been combined with data assimilation techniques such as Kalman Filters for forecasting [4]. However, approaches that simulate the interactions between surface water and groundwater using data assimilation techniques are not common. The lack of such systems is related to the fact that modeling a highly dynamic hydrological system with connected surface water and groundwater bodies is computationally very demanding. Moreover, a mechanism that automatically and continuously assimilates newly available data, updates the model in real-time, and provides predictions based on the best possible description of the current state is required. Such a mechanism significantly increases the computational demand. A vision of a new system that tackles the assimilation of environmental data in real-time, using a wireless mesh network, a cloud infrastructure, as well as a code that allows the simulation of both surface water and groundwater bodies is given by *Kropf et al.* [5].

Wireless networks connected to a cloud-based database provide an optimal solution for gathering data in remote locations and making such data available in real-time through the Internet [2], [5]. For environmental systems that require the simulation of surface water and groundwater, HydroGeoSphere [6] (HGS) presents an advanced solution, allowing for the physics-based simulation of interactions and feedback mechanisms between the two compartments [7]. The model parameters employed in HGS need to be calibrated in order to adequately represent a given environmental system. So-called data assimilation systems provide an alternative to conventional model calibration systems: they allow the sequential update of system states and model parameters whenever new data becomes available, thus guaranteeing a continuous improvement of predictions. The Ensemble Kalman Filter (EnKF) [8] provides an optimal data assimilation

mechanism for conjunctive use with HydroGeoSphere and environmental data, because it allows quantifying the uncertainties of predictions. The EnKF is already commonly applied in meteorological as well as ocean circulation simulations, but it has only recently been adapted to hydrogeological simulations [9], [10]. Like any Monte Carlo based approach, the EnKF requires a high number of simulation runs (or "realizations") to generate statistically representative probability density functions of the predictions [8]. This means that for every prediction, an EnKF-based system requires the simulation of approximately 100 realizations, out of which all individual realizations differ just slightly in terms of initial system state and/or parameter values. The prediction provided by an EnKF-based simulation is then represented by the statistical moments of the ensemble of realizations. As a high number of simulations is required, and as data assimilation techniques such as the EnKF allow continuous re-adjustment of system states and re-calibration of model parameters whenever new data becomes available, such modeling systems are ideally suited for parallelization. Notice that in our case every realization on an ordinary desktop machine requires at least 3-4 CPU hours, so the total workload is measured in approximately 400 CPU hours (16 days) for one prediction and model re-adjustment iteration. We believe that these type of computations should be ran as a cloud service, because a cloud can provide high numbers of parallel computing units; sequential execution on a single machine is impossible due to the enormous workload and hundreds of required CPU hours. This paper aims to implement a near real-time (with updates every few hours), cloud-based solution for hydrogeological simulations that are computationally very expensive, but easily parallelizable. Here, we provide a description of the fine grained architecture of a system that allows the use of real-time data in conjunction with HydroGeoSphere. The system is developed for the Emmental in Switzerland[1] in order to optimize groundwater abstraction[2] next to the River Emme.

The paper is organized in the following way: Section II provides a generic cloud computing architecture for environmental modelers coupled to real-time environmental monitoring. In Section III, we use the architecture provided to support the HydroGeoSphere application in the cloud. Section IV discusses implementation details of our solution and provides preliminary results. We conclude in Section V.

[1]The Emmental in German literally means the Valley of River Emme. The Emmental is a pre-alpine river catchment (about 200 km$^2$ wide) located in central Switzerland.

[2]Water pumping rate.

## II. General architecture

In order to comply with the requirements imposed by the EnKF-based hydrogeological modeling (c.f., Section I), we specify a cloud-based computing system. Figure 1 presents its architectural overview. It consists of two main parts: i) a Wireless Sensor Network [5] for environmental monitoring and ii) a cloud-based computational service for real-time environmental modeling.

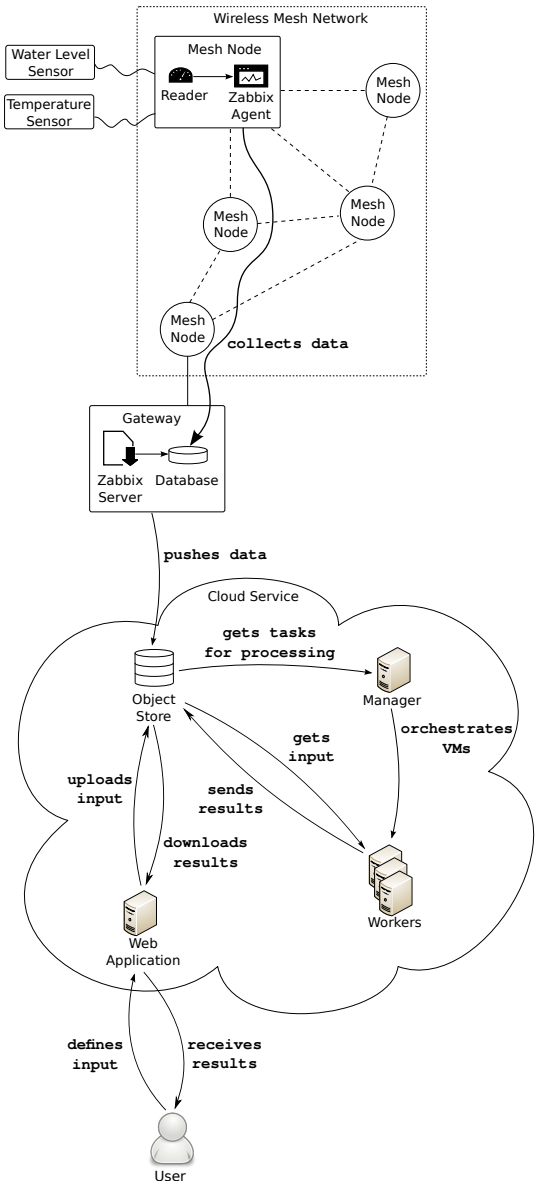


Fig. 1: General Architecture

### A. Wireless Mesh Network

Every environmental simulation requires data harvesting over a geographical area concerned. Due to recent progress in environmental sciences, real-time modeling has received significant attention over the

years. Real-time modeling requires a constant flow of data between sensor nodes located somewhere in the field and data-centers. Herein, we argue that a classical wireless mesh network equipped with sensors is beneficial for environmental researchers, because it is not restricted to the reception of mobile telephony such as GSM, and operates over long distances (e.g., IEEE 802.11 can support links of around 10 km). Also, wireless communication is significantly less expensive than any wired one in terms of installation costs; moreover, wired technology may be impossible to set-up in remote locations such as the Emmental. Lastly, wireless stations are rather portable; the process of network reorganization or relocation to another area is cheap, easy, and fast.

We deployed portable stations in the Emmental in Switzerland. Each station brings a communication unit to allow for real-time data communication. We chose IEEE 802.11 as our basic link component. In our setup, every node acts as a relay for multi-hop data forwarding. We developed a robust TCP/IP-based Wireless Mesh Network (WMN) allowing for multi-hop communication among all wireless stations because of i) careful selection of installation procedures, ii) appropriate choice of hardware components (wireless cards, directional antennas of high gain, motherboards, batteries, solar panel), iii) channel allocation schemes, and iv) auto-configuration mechanisms which include dynamic routing protocols such as Optimized Link State Routing (OLSR) and IEEE 802.11s.[3] Also, some nodes act as environmental stations; they gather information about system states, i.e., temperature, water level, and pressure, at specific geographical locations through tailored sensors such as thermometers, barometers, etc., attached to the Universal Serial Bus (USB) or Serial Ports.

### B. Hardware Platform

A node has to be capable of forwarding traffic and gathering data from sensors attached. This requires a careful selection process of the nodal hardware platform. We decided to use Alix3D2 of PCEngines. Advantages of the Alix board include low price, small size, limited power consumption, and an i386-compatible processor on-board. This comes with the cost of a little computational capacity (500 MHz CPU, 256 MB RAM), but still allows us to run a general purpose operating system such as Linux, which is able to perform TCP/IP mesh networking and environmental monitoring (c.f., Subsection II-C). This board has two mini PCI ports, so one can easily install two wireless adapters and up to two antennas (including MIMO antennas). When required, a directional antenna can support long distance connections and an omni-directional one provides short distance communication. Due to the small size, Alix boards can be placed in special-purpose enclosures which protect against environmental factors such as precipitation, humidity, etc., which are dangerous for electronic devices. When a station requires many (i.e., $\geq 2$) Wi-Fi interfaces, we accumulate a few motherboards in a single enclosure (various boxes of different size are required). Low power requirements of around 15 W per motherboard upon a standard operation allow us to solar power some stations where the electrical grid is unavailable. To maintain the 24 h uninterruptible operation, solar powered stations require batteries which are charged during the day-light operation and provide energy at night.

### C. Environmental Monitoring

Our studies reveal great similarities between environmental and system or network monitoring provided by Zabbix, Nagios, or SNMP which track the status of system or network components. Monitoring is implemented as a periodical query for the status of a certain unit. The received time-based information ($t$, value) is transmitted to a central database and stored as the time-based relation for future use such as network administration, management, or planning. Environmental monitoring is similar, because it also requires periodical information about the environmental system state[4] (i.e., periodical measurements through sensors). Received data shall be stored for example in a hydrogeological database. In such a case, the hydrogeological modeler accesses the database to get information required for providing hydrogeological forecasting on the system state in the near future.

We integrated Zabbix [11] with our mesh network. Zabbix fully implements our requirements. It is a client-server infrastructure for remote monitoring, which uses TCP/IP for the client-server communication. As a long lasting project in the open source community, it serves a large number of users. Moreover, it is extensively tested and well documented software. It therefore allows us to quickly configure all the necessary operations required by hydrogeomonitoring.

We have a Zabbix agent installed on every node in the network. It responds to monitoring queries initiated by the Zabbix server, which in our case resides on the Gateway. Normally, a Zabbix server reads out predefined parameters (e.g., traffic on interfaces), but it also supports user-defined commands to monitor user-specific hardware components. This is good, because we implement "drivers" as remote commands to support every currently deployed sensor hardware.

[3] In some cases, we provide redundant connectivity to improve performance of our network, e.g., when a link fails, our dynamic routing mechanisms switch to backup connections.

[4] "System state" should be understood as in physics, i.e., temperature, pressure, precipitation rate, etc.

Every node in the network possesses drivers to all deployed sensors. When a sensor is physically installed on a given node, we configure the Zabbix server to collect environmental information from the specific resource e.g., node, bus (i.e., USB, serial port), through a specific driver (remote command). This allows us to periodically check environmental sensors and transport readings to the gateway. For future use, the data are stored in the environmental database (c.f., Figure 1).

### D. Computational Cloud Infrastructure

Generally, running environmental simulations is computationally expensive. New methods in hydrogeological modeling, such as the real-time EnKF, amplify workload even further by simultaneously running a large number of differently parameterized models. As models gradually deviate from the real system with time, the models require initial system states to be adapted and parameters to be recomputed, in order to sustain providing accurate predictions. This is a huge effort, which requires enormous computational power [12]. We therefore decided to migrate environmental modeling into the cloud environment. In the cloud, the end-user can run a large number of parallel models, whereas one model occupies a single working machine (VM). This would then allow us to compute many models in a reasonable time and therefore use them to provide system state predictions of good quality. It is worth noting that the EnKF is easily parallelizable, because it only requires distributing completely independent workers among VMs.

Initially, when the service does not exist, the Infrastructure as a Service (IaaS) cloud plays the key role as it manages low-level cloud infrastructures. The longer term benefit is in assisting researchers to leverage IaaS architectures and to help them develop Platform as a Service (PaaS) and Software as a Service (SaaS) services to support their own communities. We use the Swiss Academic Compute Cloud (SwissACC) cloud infrastructure [13] based on OpenStack, which provides a highly scalable distributed computing infrastructure for academic research. The OpenStack platform provides us with the API to develop a highly scalable distributed computing infrastructure. In the cloud, we have deployed a leader machine, i.e. the Manager, responsible for orchestrating all other elements. The Manager spawns working VMs for newly launched computing instances and provides monitoring functions among VMs. The end-user only has a top-level view of the whole system and communicates with the Manager through the web interface. As cloud data storage we use an AWS S3 compliant ObjectStore[5,6]. The ObjectStore is used for storing input, output, and other temporary states used for the communication among distributed units in the cloud. This general architecture allows us to run generic environmental applications such as the EnKF, in which workload can be easily distributed among several parallel working instances. In Section III, we provide a detailed description of the system in which the HGS application is deployed on workers and used for hydrogeological modeling.

[5]http://aws.amazon.com/s3/

[6]RADOS REST interface used by the SwissACC ObjectStore.

## III. The HydroGeoSphere Use Case

The previously described top-level architecture matches the needs of the hydrogeological application, because multiple independent instances of HydroGeoSphere can be distributed among cloud workers (VMs). In our case, the Ensemble Kalman Filter shall be used to compute a large number of environmental simulations for the Emmental region. Subsection III-A provides the work-flow diagram of our system, in which we concentrate on interactions among distributed elements, while in Subsection III-B we provide the description of the data flow including descriptions of input and output.

### A. Work-flow diagram

Our cloud-based solution contains 3 main modules of different functionalities (c.f., Figure 2).

*a) User Interface:* the user interface allows end-users to browse and download available computed models and upload new models for processing.

*b) The Object Store:* exposes the S3 interface; it is a container wherein both input models and computed results are stored. As there is no direct interaction between the user interface and the computing unit, the object store is also a synchronization point between these two.

*c) The Cloud Task Manager (CTM):* it is responsible for processing new models uploaded to the Object Store. It performs a broad range of operations such as spawning VMs, providing input references, monitoring currently running VMs, or shutting down machines. The CTM periodically queries (e.g., every 1 minute, which should be insignificant in comparison to the model run of around 3-4 hours) the ObjectStore for new models uploaded by the user. When new models are found, the CTM launches new simulations on available or newly created VMs. The CTM supervises progress of all running simulations as well as the life-cycle of the corresponding VMs. When a new simulation is scheduled for processing, the CTM searches for an available HGS Virtual Machine (HGSVM). If no HGSVM is available, the CTM starts a new one (if possible due to resource constraints) by using the HGS Appliance Image (e.g., Linux flavor with the HGS binary and appropriate configuration for the cloud use). The HGSVM initialization process brings an

additional delay, however, it is still insignificant in comparison to the application run-time. Finally, the CTM starts a new HGS instance on the available HGSVM. The simulation instance is provided with the URI (i.e., a reference to the ObjectStore) of the model scheduled for processing. When a simulation has been completed and there are no further simulations to run, the corresponding HGSVM can be terminated. To implement the CTM, we choose GC3Pie[7] which provides a cloud orchestrating logic. Due to the fact that Python is a high-level programming language, a robust CTM can be quickly deployed. We also want to keep the project implementation homogeneous in terms of necessary programming languages to possibly re-use already implemented modules. We therefore use Django—a Python framework to implement the web interface. As an advantage, we can easily share the ObjectStore Interface between the CTM and web interface.

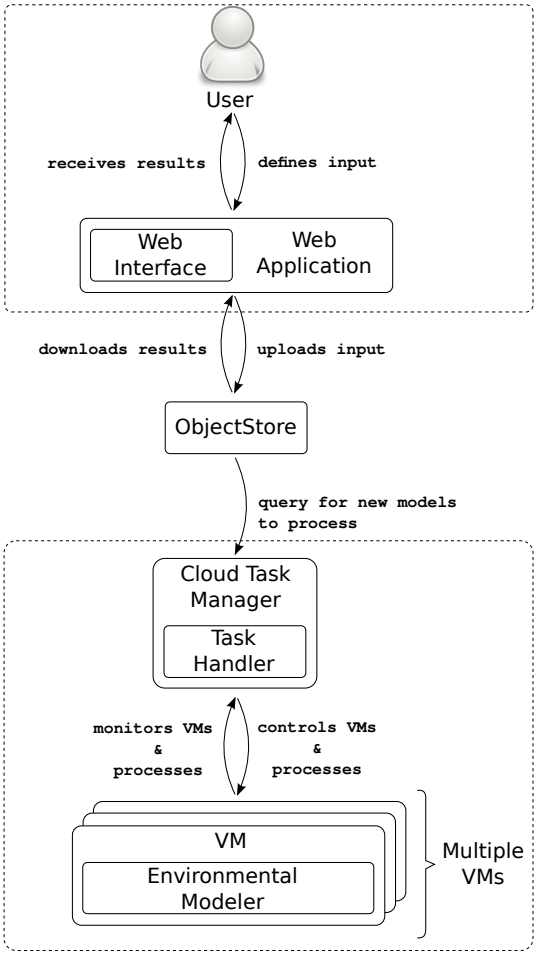


Fig. 2: Interactions

[7]GC3Pie is a framework implemented in Python and developed by the Grid Computing Competence Center (GC3) of the University of Zurich.

### B. Data Distribution Diagram

In Figure 3, we present the data distribution diagram of our distributed environment. To compute hydrogeological models, one requires various data coming from both static and dynamic resources. The terrain discretization model is fixed for a certain location, but environmental measurements are dynamic. Some environmental data is given through our sensing mesh, but other information is provided by external sources (e.g. FOEN[8] can deliver temperature and precipitation for a given region).

The end-user prepares input for a given task, which should then be computed by the HGS modeler. Some input should be assembled on-the-fly by specific data interfaces able to collect information from external sources (such as sensing mesh, FOEN, etc.). Our implementation of the dynamic interfaces is, however, still in progress, and currently the user has to manually prepare input by accessing content of the environmental database on the Gateway, copying the environmental data, preparing configuration files for the HGS application, and finally compressing all input as a compressed archive (.tar.gz). Then the user uploads the compressed archive containing the input files through the web interface to the ObjectStore. Notice that the CTM periodically looks for new tasks in the ObjectStore. When a new input is discovered, the CTM launches a new worker. The executing worker is provided with the link to the input file. It then accesses the ObjectStore, downloads the input file, uncompresses the archive, and starts a new instance of HGS. Once the work is completed, the worker uploads a compressed file with the computed model to the ObjectStore. Finally, the user can list the output bucket of the ObjectStore through the web interface and download requested result archives.

## IV. Results

Currently, we are running a hydrogeological sensing mesh in the Valley of the River Emme. The mesh network currently consists of only five nodes, among which three carry environmental sensors. For over a year now, environmental data collected by sensor nodes is uninterruptedly transmitted through the Zabbix infrastructure to the Gateway. We plan to significantly increase the size of our sensing mesh in the near future. The installation of new nodes should be straight forward due to auto-configuration mechanisms deployed on our nodes (c.f., Subsection II-A).

We have implemented the web interface in Django and the CTM based on the GC3Pie framework. Access to the ObjectStore is provided through the S3 driver, however, to simplify our application we use

[8]FOEN is The Swiss Federal Office for the Environment.

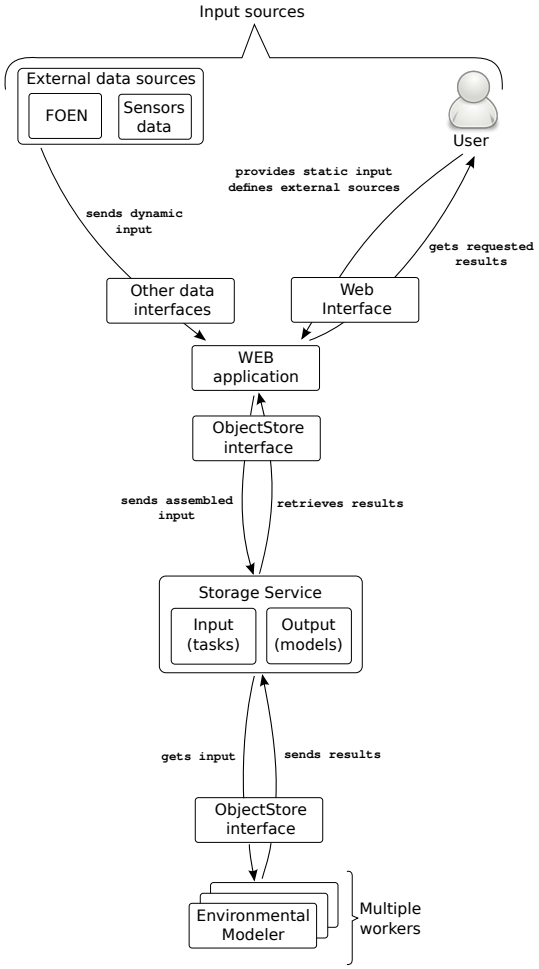


Fig. 3: Data Distribution Diagram

an existing storage back-end module for Django—Django-storages. We also adapted the HGS binary to operate on a selected cloud Linux flavor. Notice that the CTM deploys a running instance of the VM, which in our case works on the Ubuntu 12.10 Linux distribution. Such a VM contains a working copy of the HGS application, which has to be compatible with the operating system architecture on-board. Currently, we do not implement any external data sources. The end-user has to manually create a compressed archive with the HGS configuration and upload it through the web page. In the future, the input data and archive generation for HGS will be automatically done through the EnKF software. The input archive provided is installed on the ObjectStore through the S3 driver. Once every minute, the CTM searches for new input in the input bucket. When possible (no quota exceeded, enough free resources in the cloud), the CTM allocates a new VM with Ubuntu. The VM goes to the ObjectStore and downloads the input archive through S3cmd. The archive is uncompressed and the VM's working instance of the HGS application is provided with input

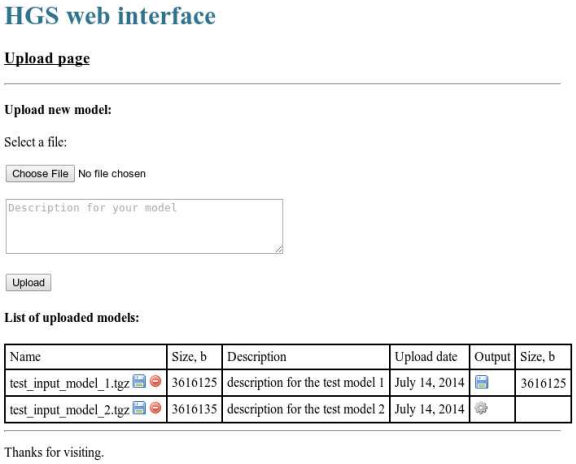


Fig. 4: Web Interface

(notice that currently, we do not perform any checks on the consistency and correctness of the input file). The HGS application starts computing and works until the task is finished. Then the compressed HGS results are uploaded to the output bucket of the ObjectStore and the VM is released or terminated. The end-user can use the web interface to list and download output of his/her computation (c.f., Figure 4). Currently, our infrastructure allows the user to deploy a large number of simultaneous HGS computing instances. The implementation of the EnKF requires more effort as all models have to be automatically verified against the measurements of the real system's state. Furthermore, predicted system states and model parameters shall be regularly adjusted over time, and statistical moments have to be computed over the ensemble of predictions for environmental forecasting.

## V. Conclusions

Firstly, we designed a modular architecture that provides a quick integration of an easily parallelizable environmental computing application with a cloud computing environment. Secondly, we showed that the HydroGeoSphere application, which is used in the domain of hydrogeology, can be successfully run in a distributed way to the benefit of the hydrogeological community. The cloud proved that it can quickly distribute the load of many simultaneously computed models, thus it allows for computationally expensive modeling. As a future work, we shall implement on-the-fly integration of dynamic data sources and the environmental Ensemble Kalman Filter in the cloud environment. In such a solution, the main interest lies in the full integration of cloud infrastructure services (both computing and storage) for a high-throughput hydrological application with the objective of allowing for near real-time processing of sensor data.

## REFERENCES


[1] G. Badawy, A. Sayegh, and T. Todd, "Solar Powered WLAN Mesh Network Provisioning for Temporary Deployments," in *IEEE Wireless Communications and Networking Conference (WCNC)*, Mar. 2008, pp. 2271–2276.

[2] D. Wu and P. Mohapatra, "QuRiNet: A wide-area wireless mesh testbed for research and experimental evaluations," in *Second International Conference on Communication Systems and Networks (COMSNETS), 2010*, Jan. 2010, pp. 1–10.

[3] E. Basha and D. Rus, "Design of early warning flood detection systems for developing countries," in *International Conference on Information and Communication Technologies and Development, 2007. ICTD 2007.*, Dec. 2007, pp. 1–10.

[4] P. J. Smith, D. Hughes, K. J. Beven, P. Cross, W. Tych, G. Coulson, and G. Blair, "Towards the provision of site specific flood warnings using wireless sensor networks," *Meteorological Applications*, vol. 16, no. 1, pp. 57–64, 2009. [Online]. Available: http://dx.doi.org/10.1002/met.130

[5] P. Kropf, E. Schiller, P. Brunner, O. Schilling, D. Hunkeler, and A. Lapin, "Wireless Mesh Networks and Cloud Computing for Real Time Environmental Simulations," in *Recent Advances in Information and Communication Technology*, ser. Advances in Intelligent Systems and Computing, S. Boonkrong, H. Unger, and P. Meesad, Eds. Springer International Publishing, 2014, vol. 265, pp. 1–11.

[6] R. Therrien, R. G. Mclaren, E. A. Sudicky, and S. M. Panday, "HydroGeoSphere: A Three-dimensional Numerical Model Describing Fully-integrated Subsurface and Surface Flow and Solute Transport," Waterloo, Ontario, Canada, Tech. Rep., 2007.

[7] P. Brunner and C. T. Simmons, "Hydrogeosphere: A fully integrated, physically based hydrological model," *Ground Water*, vol. 50, no. 2, pp. 170–176, 2012. [Online]. Available: http://dx.doi.org/10.1111/j.1745-6584.2011.00882.x

[8] G. Evensen, "Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics," *Journal of Geophysical Research: Oceans (1978-2012)*, vol. 99, no. C5, pp. 10 143–10 162, 1994.

[9] H. J. Hendricks Franssen and W. Kinzelbach, "Real-time groundwater flow modeling with the Ensemble Kalman Filter: Joint estimation of states and parameters and the filter inbreeding problem," *Water Resources Research*, vol. 44, no. 9, 2008. [Online]. Available: http://dx.doi.org/10.1029/2007WR006505

[10] W. Kurtz, H.-J. Hendricks Franssen, and H. Vereecken, "Identification of time-variant river bed properties with the ensemble Kalman filter," *Water Resources Research*, vol. 48, no. 10, 2012. [Online]. Available: http://dx.doi.org/10.1029/2011WR011743

[11] A. Vladishev, "Zabbix home page—an Enterprise-Class Open Source Distributed Monitoring Solution," http://www.zabbix.com/, 2001.

[12] H.-T. Hwang, "Development of a Parallel Computational Framework to Solve Flow and Transport in Integrated Surface-Subsurface Hydrologic Systems," *PhD. Thesis, University of Waterloo, Canada*, 2002.

[13] P. Kunszt, S. Maffioletti, D. Flanders, M. Eurich, E. Schiller, T. Bohnert, A. Edmonds, H. Stockinger, A. Jamakovic-Kapic, S. Haug, P. Flury, and S. Leinen, "Towards a Swiss National Research Infrastructure," in *Proceedings of the 1st international workshop on federative and interoperable cloud infrastructures, FedICI'13 organized in conjunction with Euro-par 2013*, Aug. 2013.