21st International Congress on Modelling and Simulation, Gold Coast, Australia, 29 Nov to 4 Dec 2015
www.mssanz.org.au/modsim2015

Performance characteristics of Source calibration service

R.M. Singh [a], L. Taylor [a], D. Penton [a], M. Stenson [a], G. Podger [a] and A. Brown [a]

[a] *Land and Water Flagship, Commonwealth Scientific and Industrial Research Organization (CSIRO)*
Email: ramneek.singh@csiro.au

Abstract: Using optimisers to calibrate hydrological models is a computationally intensive process. Most optimisation algorithms run on desktop machines, with some running on Linux clusters and a couple that run on cloud infrastructure (e.g. cloudPEST). Complex hydrological models require a relatively powerful machine and calibration runtimes vary from an hour or less, to days and sometimes weeks. Increasingly, organisations are looking to outsource provision and management of computationally intensive infrastructures. While virtualisation technology can provide similar performance to high end desktops, there are opportunities to harness parallelisation and reduce calibration times, by hosting the modelling software on the cloud infrastructure and exposing its functionality through web services. This paper investigates the practicality and performance of implementing a calibration wrapper to the eWater Source river modelling package. The Source calibration service allows user to calibrate models, where the modelling software, eWater Source, is running on the cloud and not on end user's premises.

The aim of this analysis was to compare the performance characteristics of a simple GR4J model for the Legerwood catchment using eWater Source running as desktop software versus running as a Source calibration service on the cloud. Shuffle Complex Evolution was used as the parameter optimisation algorithm for the GR4J model parameters.

The eWater Source product running as desktop software took around 4 minutes to calibrate the model whereas the Source calibration service took around 73 minutes to do the same calibration with similar results. The difference in run times can be attributed either to: 1) the chatty nature of communication between the machines running the eWater Source and the optimization algorithm; and/or 2) time inefficient implementation of SCEoptim routine from the hydromad package; and/or 3) performance bottle necks in Source's external interface which exposes eWater Source modelling capability through command prompt. Given the long simulation runtimes, the current Source calibration service fails to meet expectations of hydrological model builders for improved performance. For software implementers, we would recommend careful attention to the software architecture and performance characteristics of proposed cloud-based software implementations early in development. In this case, we anticipate future improvements to the infrastructure, or renewed effort improving the implementation would lead to a faster implementation.

Keywords: *Web services, hydrological model calibration*

1. INTRODUCTION

Calibrating hydrological models is a time and compute intensive process. The aim of the exercise outlined in this paper was to use the eWater Source hydrological modelling software as a service on the cloud to calibrate catchment models. Legerwood catchment (Figure 1) in Tasmania was used for calibration purposes and was modelled using daily Rainfall-Runoff (RR) model Ge´nie Rural a` 4 parame`tres Journalier (GR4J) (Perrin, Michel and Andreassian, 2003).

GR4J model takes inputs of daily rainfall and potential evapotranspiration and gives an output of daily runoff. Calibrating a GR4J model involves choosing a GR4J parameter set that best fits the observed runoff for a given period. Shuffle Complex Evolution (SCE) optimization (Duan et al., 1992; Duan et al., 1993; Duan et al., 1994) routine was used to calibrate the model parameters for the GR4J model. The objective of the calibration was to maximise Nash-Sutcliffe efficiency (NSE) (Nash and Sutcliffe, 1970) of model predictions compared to the observed data. SCE optimization routine was chosen because it is frequently used in water resources sector for hydrological model calibration and evaluation.

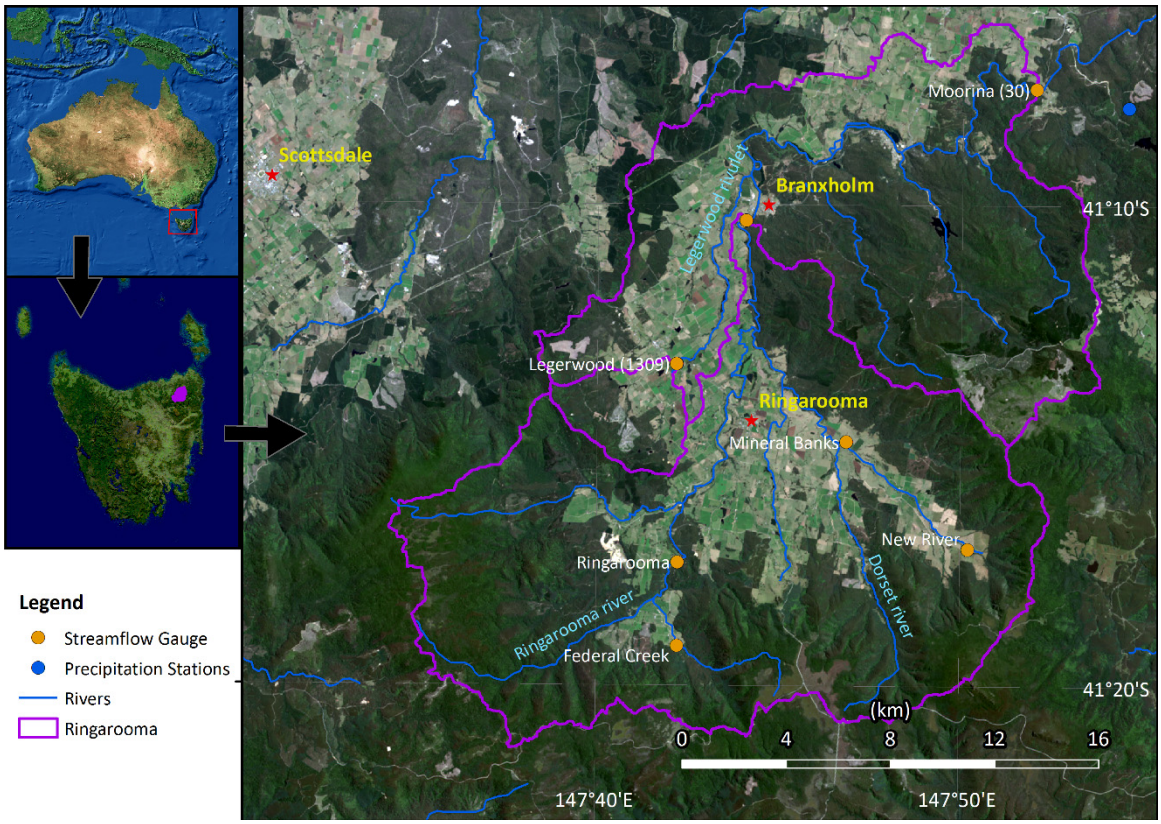

Figure 1. The Source model represented the runoff of the Legerwood catchment in North East Tasmania.

2. PROJECT SETUP

2.1. Overview

eWater Source (Source) is a desktop hydrological modelling software written using the Microsoft .net framework. Modelling Services Interface (MSI) exposes the modelling capability of Source through RESTful calls over the web (Stenson et al., 2014; Leighton et al., 2011). Figure 2 shows the architecture of MSI and potential different scenarios making use of RESTful calls to run Source on the cloud. Having the Source modelling ability available on the cloud allows researchers and modelers in developing countries, who might not have appropriate hardware resources, to run simulations. The results of such simulations could be delivered by email or made downloadable via a webpage. Once a simulation on the web is started, the client machine which instantiated the simulation need not stay powered on as the actual computations take place on a cloud server.

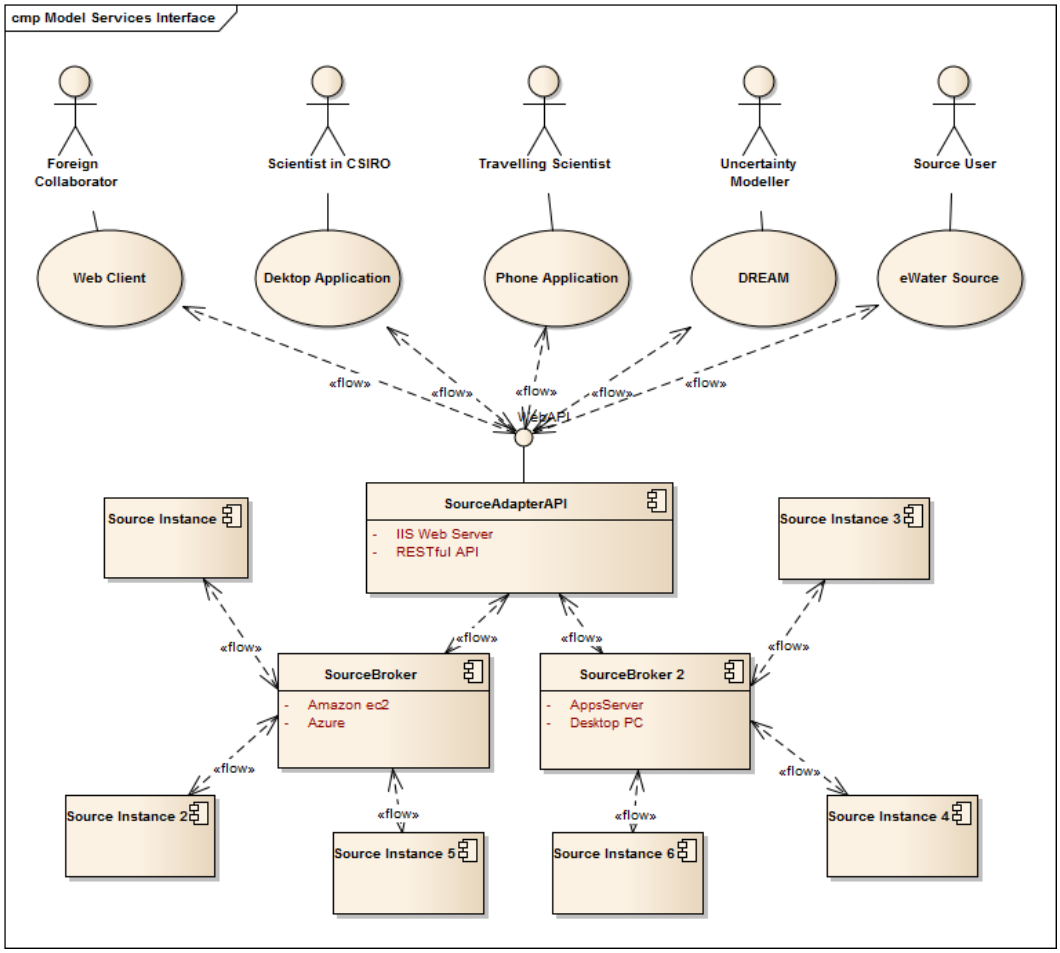

Figure 2. The Source Modelling Services Interface (MSI) component diagram (from Stenson et al. (2014). An Application Of Services Based Modelling Paradigm To The Hydrologic Domain Using eWater Source)**.**

SourceAdapter API component of the MSI, which acts as the gateway to Source modelling functionality, is an ASP.net application hosted on IIS web server. The SourceAdapter server is associated with one or more SourceBroker nodes which can either be virtual or physical machines. Each SourceBroker node runs the SourceBroker console application which starts the SourceBroker Windows Communication Framework (WCF) service to communicate with the SourceAdapter. SourceBroker service manage individual Source instances running from the command line interface of Source, also called Source External Interface.

On top of SourceAdapter API RESTful service layer, which runs on a windows machine, two additional layers were written:

1) Middle layer: Written in R, this layer does the calibration of the model exposed through SourceAdapter API. For the optimization routine, SCEoptim from the hydromad R package (Andrews, 2011) was used along with NSE objective function which was also encoded in R. This layer runs on a Linux Ubuntu machine. As explained in section 2.2 (Cloud calibration workflow), there is bi-directional communication between this layer and the SourceAdapter API layer.

2) User interface layer: For this layer, a web application was written using Meteor, which is a JavaScript framework built on top of node.js. This web application allows user to make selections around the calibration process, such as defining parameter ranges for GR4J parameters and configuring the SourceAdapter API setup. This layer currently runs on a Linux machine, though meteor is a cross platform product. As explained in section 2.2 (Cloud calibration workflow), there is bi-directional communication between this layer and the Middle layer.

Figure 4 shows the bi-directional communication between these 3 different layers. The architecture of the project allows for the addition of different optimizers to the cloud calibration system. As shown in Figure 3, at the moment a user can select one of two optimizers implemented to support cloud calibration through the web interface: SCE and DREAM (Vrugt, 2009).

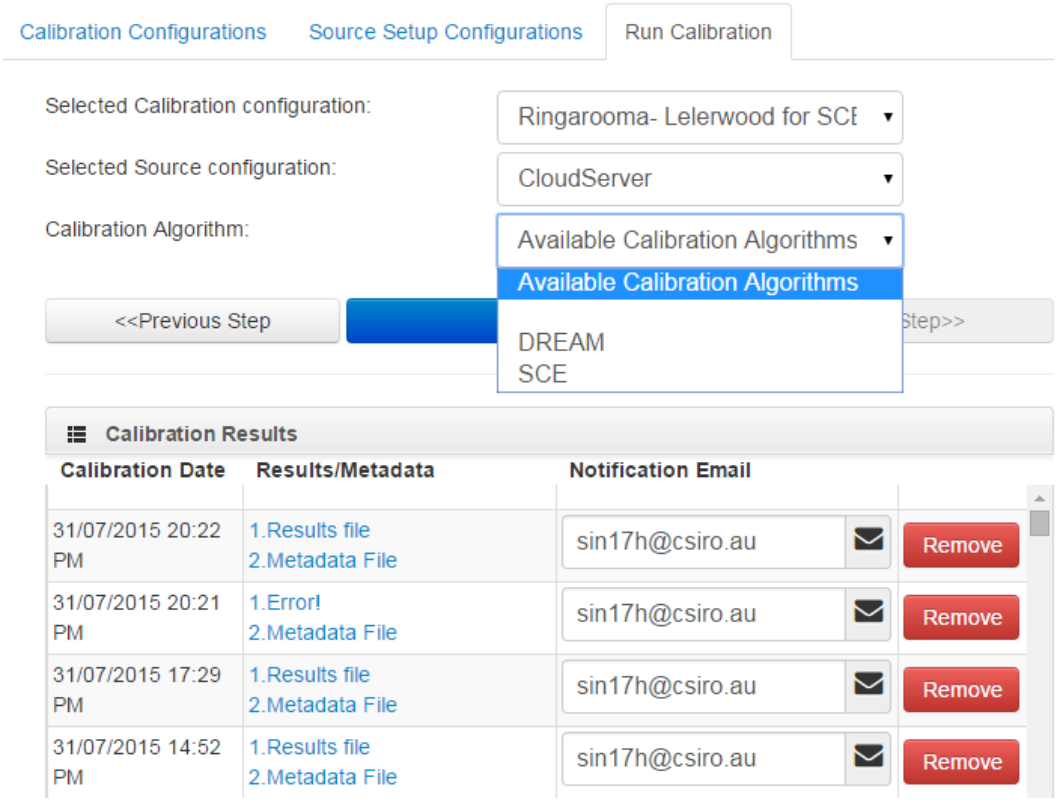

Figure 3. Available calibration algorithms

Source was used to construct a GR4J model for Legerwood catchment which was then used for calibration through both the cloud calibration setup as well as local Source calibration.

2.2. Cloud calibration workflow

To calibrate a model on the cloud the user has to open the web application in a web browser. This web application allows the user to setup a calibration, choose a cloud server setup and then run a calibration using one of the available optimizers. The configuration data defined on the web page can also be imported into the system via JSON files (Crockford, 2006). For this experiment, SCE was selected as the optimization algorithm. An example calibration configuration for Legerwood catchment is shown in Figure 5.

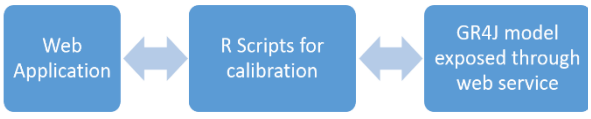

Figure 4. Communication flow between components

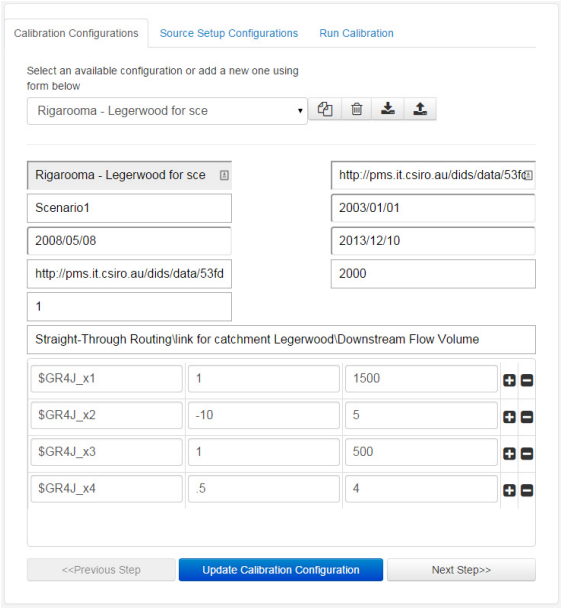

Figure 5. Calibration web application

On running a calibration through the web application, the execution control is transferred to Meteor server side code where the user selected options are serialized to lightweight JavaScript Object Notation (JSON) format and a new child process is spawned running the appropriate R scripts depending on the user selected calibration algorithm. These R scripts in turn start the calibration process and make RESTful calls to the SourceAdapter API to run the GR4J model. To and fro communication takes place between all the three different layers for various purposes.

Once the calibration has started, the SCEoptim routine reports timing information about the execution back to the web application every 10 iterations. The web application uses this timing information to make an estimate about the time it would take to complete the calibration and then shows it to the user. SCEoptim uses the NSE objective function to calibrate the GR4J model exposed through SourceAdapter API. For each iteration, new set of parameter values is chosen by SCEoptim and the values are passed to the GR4J model through a web service call. And when the GR4J model has completed a single run using the parameter values supplied, the result time series is passed back to R layer. An NSE value is calculated based on the returned results. This two way communication continues until we meet the objective of minimizing the difference observed results and calculated results. Figure 6 shows a high level sequence diagram outlining the control flow between different components of the setup.

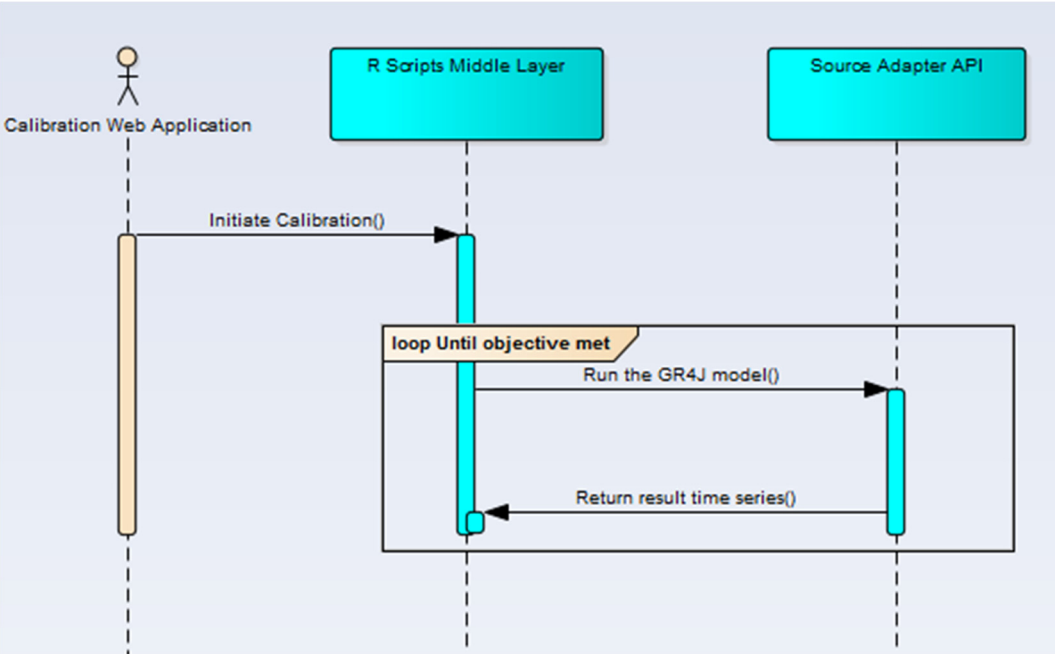

Figure 6. Sequence diagram showing the flow of communications

2.3. Results

Running the calibrations on cloud infrastructure compared to on a local machine was much slower. Using the Legerwood catchment model, Source running as a desktop application took around 4 minutes to complete the calibration in 1082 iterations. The same model when calibrated using the cloud setup took around 73 minutes and 1270 iterations to calibrate. Slight differences in the calibration setup led to a different NSE value and number of iterations. Table 1 summarizes these results and Figure 7 and 8 shows the convergence of objective function score towards a solution for both the local and cloud runs. We have not thoroughly investigated the reasons behind why it took so long to calibrate on the cloud due to time restraints, however some of the reasons could be:

1. Chatty communication between Source adaptor API, running on a windows machine and SCEoptim, running on a Ubuntu Linux machine, is causing the calibration to run slow as for each iteration the result time series is sent back over the network to SCEoptim where an objective score is calculated using the result time series and the observed time series and based on that score either:
    a. The calibration is stopped if it meets the threshold defined or

b. The Source adaptor API is again called with a new set of GR4J parameter values chosen by SCEoptim implementation.

If chatty communication is the cause for slow calibration times, then there could be significant improvements to calibration runtimes if model simulation runtimes are greater in magnitude to communication cost involved thereby minimizing the effect of back and forth communication.

2. Source External interface, on top of which the SourceAdapter API has been developed, has performance bottlenecks which cause the Source model to run slower that usual. Profiling the Source external interface code might help us analyze the perfromance bottlenecks.

Although unlikely, the SCEoptim implementation of SCE from the hydromad package may not be as efficient in terms of time taken to calibrate. If this is the case, using another implementation of SCE algorithm might show better results. Some preliminary profiling of execution times in various components of the system for a single run gave these results:

1) Total time to complete the calibration: 73 minutes.
2) Total time spent in the R function that SCEoptim is trying to optimize: 72.14 minutes
3) Total time spent in SourceAdapter API code for setting the metaparameters for each run and then running the model: 55.55 minutes

Now SourceAdapter API itself is just a delegator which forwards the metaparameters for each run to the Soure instance running through command prompt. So 76% of the time is spent running the model through the Source external interface. This supports our hyothesis that there might some bottlenecks in the interface/internals of eWater Source system which need to be analyzed.

Table 1. Results

Implementation	Time taken	No. iterations	x1	x2	x3	x4	NSE
Source	4 mins	1082	884.50	-3.94	157.90	0.66	0.849
SCEoptim	73 mins	1270	934.17	-3.86	152.61	0.50	0.850

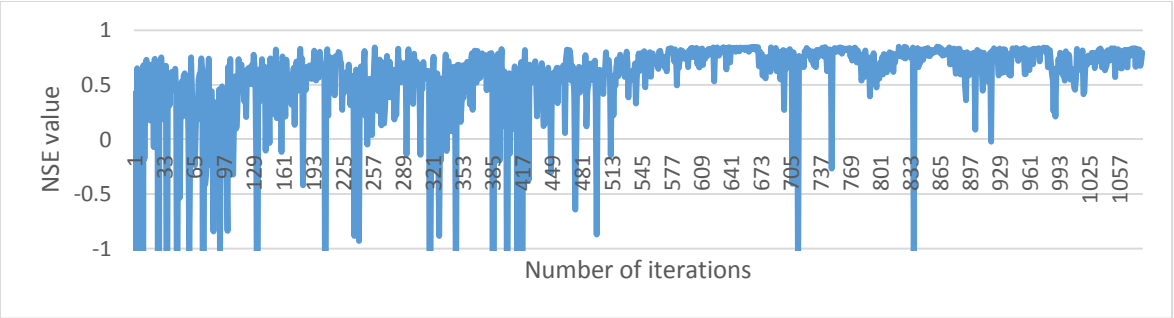

Figure 7. Convergence of NSE score for local Source run

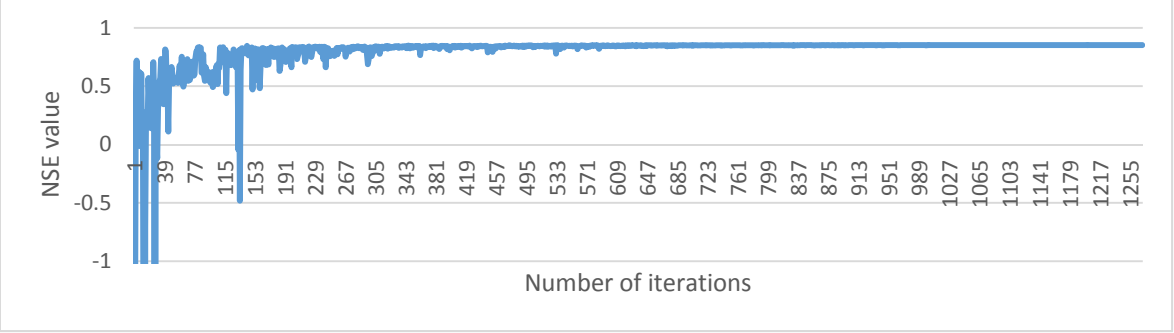

Figure 8. Convergence of NSE score for Source web service run

3. DISCUSSION AND CONCLUSIONS

Source running locally took 4 minutes to calibrate the same GR4J model which took 73 minutes to calibrate on the cloud. So the performance of the cloud setup is a lot worse than running a Source calibration locally. Although the reasons behind the slow performance of cloud setup have not been analyzed yet, there could be several reasons which explain it. Given the time constraints, we were not able to test the possible reasons which lead to making the cloud setup of Source slow for running calibrations. Future work planned for the project involves testing the interfaces/internals of various components in this setup and profiling the code to come up with bottlenecks that exist in the system. Effort needs to be spent to improve the performance and then benchmark the calibration performance of Source calibration service vs Source run locally. For future implementers of such cloud-based systems, our advice would be to test the interfaces of various components from a performance perspective as they develop the system. Doing so might help in some circumstances wherein just changing the architecture of the setup can mitigate some performance issues.

ACKNOWLEDGMENTS

The Sustainable Development Investment Portfolio (SDIP) is funded by the Australian government's Department of Foreign Affairs and Trade (DFAT). This work was funded by SDIP and partly by CSIRO strategic funding.

REFERENCES

Fienen, M. N., Kunicki, T. C., & Kester, D. E. (2011). *cloudPEST-A python module for cloud-computing deployment of PEST, a program for parameter estimation* (No. 2011-1062). US Geological Survey.

Duan, Q., Soorooshian, S. and Gupta, V., 1992. Effective and efficient global optimization for conceptual rainfall-runoff models. *Water Resourc. Res.* 28 (4), 1015-1031.

Duan, Q., Gupta, V.K. and Sorooshian, S., 1993. A Shuffled Complex Evolution approach for effective and efficient global minimization. *Journal of Optimization Theory and its Applications*, 76 (3), 501-521.

Duan, Q., Sorooshian, S. and Gupta, V.K., 1994. Optimal use of the SCE-UA global optimization method for calibrating watershed models. *Journal of Hydrology*, 158 265-284.

Perrin, C., Michel, C. and V. Andreassian, (2003), Improvement of a parsimonious model for streamflow simulations. *Journal of Hydrology*, 279, 275–289.

Nash, J. E. and J. V. Sutcliffe (1970), River flow forecasting through conceptual models part I -A discussion of principles, *Journal of Hydrology*, 10 (3), 282-290.

Stenson, M., Penton, D., Leighton, B., Car, N., Bai, Q., Singh, R., Perraud, J.M., and Bridgart, R. (2014). An Application Of Services Based Modelling Paradigm To The Hydrologic Domain Using Ewater Source. 11th International Conference of Hydroinformatics (HIC2014), New York, USA, 17-21 August 2014. Paper 1145.

Leighton, B., Manser, P., Penton, D., Shoesmith, J., Stenson, M., & Vleeshouwer, J. (2011). Exposing a hydrological simulation model on the web. In *19th International Congress on Modelling and Simulation (Modsim2011)* (pp. 1230-1236).

Andrews, F.T., Croke, B.F.W., and Jakeman, A.J. (2011). An open software environment for hydrological model assessment and development. *Environmental Modelling and Software*, 26, 1171–1185.

Crockford, D. (2006). The application/json Media Type for JavaScript Object Notation (JSON). Memo, The Internet Society. Retrieved July 6, 2011, from http://www.ietf.org/rfc/rfc4627.txt?number=4627

Vrugt, J. A., Ter Braak, C. J. F., Diks, C. G. H., Robinson, B. A., Hyman, J. M., & Higdon, D. (2009). Accelerating Markov chain Monte Carlo simulation by differential evolution with self-adaptive randomized subspace sampling. *International Journal of Nonlinear Sciences and Numerical Simulation*, *10*(3), 273-290.