8-1-2014

An Application Of Services Based Modelling Paradigm To The Hydrologic Domain Using Ewater Source

Matthew Paul Stenson

Dave Penton

Ben Leighton

Nicholas J. Car

Qifeng Bai

See next page for additional authors

11th International Conference on Hydroinformatics
HIC 2014, New York City, USA

AN APPLICATION OF SERVICES BASED MODELLING PARADIGM TO THE HYDROLOGIC DOMAIN USING EWATER SOURCE

MATT STENSON (1), DAVE PENTON (2), BEN LEIGHTON (3), NICK CAR (1), QIFENG BAI (2), JEAN-MICHELE PERRAUD (2), ROB BRIDGART (2), DAVID LEMON (2)
(1): CSIRO Land and Water, Dutton Park, Brisbane, QLD, Australia
(2): CSIRO Land and Water, Black Mountain, Canberra, ACT, Australia
(3): CSIRO Land and Water, Highett, Melbourne, VIC, Australia

The traditional paradigm for the deployment of hydrologic based models involves the capturing and testing of model concepts and numerical consistency for robustness and accuracy, which is then distributed as binary files with or without source code. The model software is then populated with data and parameters, and run locally within the modeller's organisation, often on their own desktop. This modelling workflow is used by many organisations; however, there are several limitations and potential issues. Once the software is outside the developer's organisation they rely on the modeller to apply updates and bug fixes in a timely manner, and to correctly describe the model version used for reporting. The developer also loses control of the quality and suitability of the input data for a particular application of the model. With the more prevalent access to high bandwidth internet and flexible computing infrastructure there is an increased opportunity to better control model access through the exposure of modelling functionality through web services. As well as giving the developer tighter control over model versioning and IP, it also allows the closer coupling of the model to both the data sources and computational resources, which is especially beneficial to multi run use cases such as uncertainty analysis and calibration, where the ability to easily scale to many model instances is of most value. The eWater Source modelling system is an important use case for Australia's hydrologic community, and provides a rich array of functionality. Source is especially suited to the services modelling paradigm as it has project load times much greater than simulation runtimes, the services based approach allows the hiding of these load times by keeping the project in memory for each instance of a Source Server. This paper investigates the use of Source service interface for providing hydrological modelling web services.

INTRODUCTION

The software that water planning model builders use for developing hydrological models (e.g. SWAT, MIKE-SHE) are deployed on local machines as desktop software or occasionally on high performance clusters that are maintained by their institutions. In this paper we test the hypothesis that recent advances in cloud computing technologies will provide an alternative method for software development of hydrological models. The case study for this paper is the application of parameter uncertainty methods (DREAM-MCMC) to the eWater Source hydrological model. The results of a local implementation using generic techniques show that the improvements are non-linear, but that where simulation runtimes are greater in magnitude to communication cost there can be significant improvements. Limitations in the scope of our study, lead us to conclude that more work is required to appraise the relative merits of cloud computing as a deployment platform for hydrological modelling.

RATIONAL

What is cloud computing?

Cloud computing is a somewhat vague and overloaded term. In the context of this paper it refers to a managed set of practically infinitely scalable virtual computing resources connected in real time and available on a user pays basis. These resources include but are not restricted to web servers, virtual machines, data storage and backup.

Why the cloud?

Elastic scaling: The volumes of data collected globally are increasing exponentially. In the scientific world this means an increase in either the spatial or temporal resolution of the data, or in the number of physical processes or indicators being observed. Often our ability to make use of this data is restricted by the practical constraints of computational availability. Additionally, the understanding of model results, our ability parameterise them effectively, and our ability to capture and express their inherent and compounded uncertainty relies on multiple model runs, sometimes many tens of thousands of runs. For computationally complex models, or models with large data processing components, the associated long run times become prohibitive to all but organisations with access to large computational infrastructure.

In the scientific domain, to increase the modeller's ability to undertake computationally intensive activities without reducing the scale or complexity of the problem, or resorting to expensive and time consuming optimisations, the problems were often scaled across multiple processing threads, e.g. Central Processing Units (CPU) or CPU cores, or across multiple computers, e.g. computational clusters. Adding more CPU's cores is known as vertical scaling and while offering an easy solution especially from the perspective of communication between processing threads, becomes expensive and is limited by computer design to at most many dozens of cores. Adding more computers, or virtual computers to a processing problem is known as horizontal scaling and unlike vertical scaling which has physical limits and becomes prohibitively expensive, horizontal scaling is effectively limitless and is only restricted by the ability of each computer, or node connected to the network of computers to communicate within acceptable time limits.

In a traditional scientific organisation large computational clusters attached to very fast storage are purchased, maintained and depreciated by the organisations themselves, or through

partner networks. These computing facilities are expensive to run and maintain, slow to scale and upgrade, and must be run at near capacity to realise return on the investment. The model is inherently inflexible. Another major advantage of cloud based computing is the ability to de-scale as required, relinquishing compute resources as they are no longer needed, and therefore resources that are no longer paid for by the user. This user pays model, combined with infinite scalability is at the core of cloud computing.

Collaboration: Due to real or perceived security threats arising from network access, and issues of who pays for computational resources, collaboration between institutions is often difficult and slow to arrange. As cloud computing is wired into the internet collaboration can occur without institutional arrangements, using standard computer security procedures. While cloud computing doesn't solve the 'who pays' issue between collaborators, it does at least move it to a neutral provider with transparent costing.

Repurposing: Scientists and modeller love to push the boundaries on everything they do, and this is especially true of computing. Having the ability to explore new techniques, software and tools at will is often at odds with corporate IT departmental policy for providing a stable and secure computing environment at a low cost. Cloud computing allows virtual services such as web servers, data storages or virtual machines to be 'stood up' at will and the resources released when no longer needed, and as the resources are internet accessed, they pose little threat to the local corporate network.

Control of IP and appropriate use: Traditionally modelling software is released at a certain version, as is to the world at large. Once it is released the IP becomes a lot harder to control, and more importantly, control over appropriate use of the model is reduced to soft recommendations in user manuals and published papers. By exposing modelling software as a service through cloud infrastructure, the publisher of the model has total control over model versioning and can more closely enforce constraints around input data and parameters, additionally the IP is contained within the model.

Modelling in the developing world: The understanding of physical processes as expressed in modelling software, and experience in applying those models to inform management decisions and policy is often directly applicable between the developed world and the developing world. For modelling software to have long term impact it needs to be embedded in local countries and their operational agencies. While the model understanding and processes are often easily transferrable, they tend to only stay current for the duration of the project facilitating the transfer. Additionally, a lack of substantial computational resourcing is common, restricting the ability to do ongoing computationally intensive tasks such as calibration and uncertainty analysis. Cloud computing allows for not only continual access to updated model versions through the publishing of modelling services, but also access to infinitely scalable computational infrastructure and storage.

Why not the cloud?

Initial costs (build new software): With few exceptions, modelling software is not setup to take advantage of the cloud for example by being exposed through web services. This means service type layers will need to be written on top of existing applications or written into a modified version of the application. Depending on the modelling application, the technologies used to develop it, and the complexity needed to be exposed this can be an expensive and complicated exercise.

Who pays: Cloud services are provided on a user pays basis. When provisioning these services an account is needed to absorb the usage charges. In modelling software as a service it

is not always clear who will pay for the cloud services, especially if the computation, storage or bandwidth usage is high. One of the main benefits of exposing models on the cloud is increased availability and usage, but the developing organisation is unlikely to want to absorb costs for other users. One solution is to have other user's setup specific instances of a modelling service for their own usage, but this is akin to the more normal modelling software distribution methods where the developers no longer control either the IP, or can guard against inappropriate usage.

MODELLING SERVICES INTERFACE

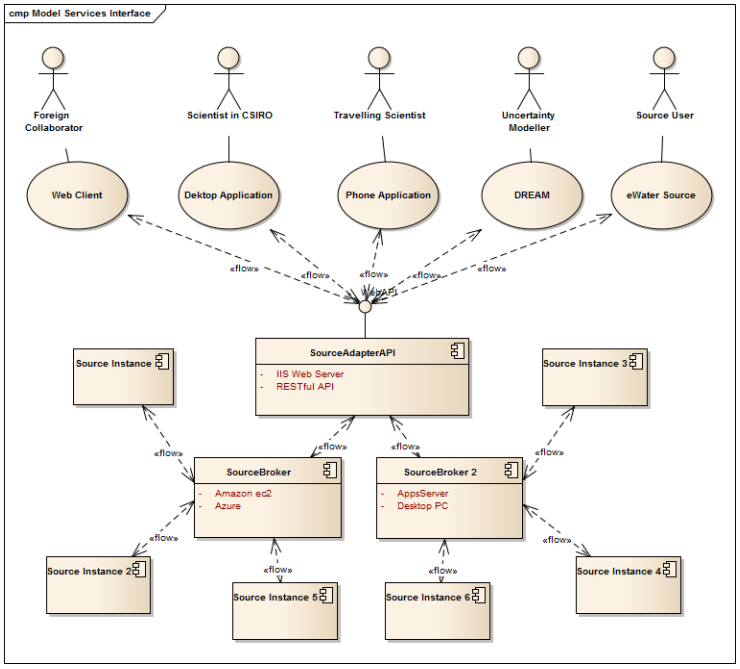
The Modelling Services Interface (MSI) (Figure 1) is a collection of web based services that together allow for the remote execution of modelling jobs. The service provides a level of abstraction between the compute nodes and the calling code, and manages all of the resources transparently from the user. It is designed to be scalable and easily deployable across remote infrastructure such as commercial clouds. It is hoped that when finalised, it will provide a consistent design pattern for exposing complex simulation models through a set of standardised RESTful calls in much the same way as a Web Processing Service (WPS) does for spatial and GIS processing jobs. A first test implementation of MSI based around the eWater Source Modelling System (Carr and Podger 2012 [1]) (Welsh, Vaze et al. 2013 [10]) with some initial results shown below in this paper. Once the design pattern for the interface has been worked out it is hoped to test it with another complex model.

The Source Modelling Service is a multi-tiered design:

- Layer 1 (AKA processing node) is the RiverSystemServicesCluster RSSC (Leighton, Manser et al. 2011[4]). It is a series of WCF services for managing individual Source Instances. There is one instance of RSSC per virtual machine, with 1-n* virtual Machines (VM) within a Source Modelling Service Job. The number of Sources Instances per RSSC is only limited by the memory available on the host VM.
- Layer 2 (AKA processing node) is a console application (SourceBroker Figure 1) that starts, stops and hosts the RSSC WCF service. There will be one per RSSC.
- Layer 3 (AKA web service) is the SourceAdapterAPI (Figure 1), a ASP.NET Web.API that exposes a number of RESTful calls to manage Source Modelling jobs, and distribute the load across multiple RSSC instances. This will be the main interface with the Source Modelling Service.
- Layer 4 (AKA Web application); This layer can and will have many views. It will leverage the SourceAdapterAPI to expose various levels of functionality to the users. This could be through a fully featured web site, a mobile device with reduced functionality, or directly in code as an interface between other software and the Source Services as shown in this paper interfacing with DREAM.

API Description

The SourceAdapterAPI is a RESTful API web services interface designed to satisfy multiple modelling web services use cases for the eWater Source modelling system. The API has several published RESTful controllers designed to make creating and managing Source Modelling jobs easier.



Using the API

The API is designed to be as flexible as possible. It can be used either by typing calls directly into a web browser, through a host web page that hides the actual RESTful calls behind buttons and other widgets such as on a data portal, or directly through code, for example when using the service to perform large computationally expensive operations such as uncertainty analysis or calibration.

Figure 1: The Source Modelling Services Interface component diagram showing the Source Adapter API as the central hub of coordination for multiple modelling jobs, each with its own n number of SourceBroker's, which in turn have n number of instances of Source, each holding a loaded project.

Basic concepts

Job: A job is a container for managing multiple Source instances with a single project file. A job can be as simple as a single Source run, or as complex as thousands of runs spread across dozens of virtual machines.

Instance: Each job contains 1-n* number of instances. These instances may be shared across many virtual machines known as nodes. Instances can be added or removed from a Job, run as a group or individually and have their parameters updated using an appropriate meta-parameter.

Meta-parameter: A unique string representing an internal Source model parameter or state that can either be recorded or changed through the modelling service. These can be used for recording outputs, or adjusting parameters for purposes such as model calibration. Meta-parameters are set up before being submitted to a SourceAdapter modelling Job through the Source GUI. A list of meta-parameters in a current Source project can be viewed by using the GET api/Instance/id?jobId=jobId command. Model output meta-parameters are requested at a Job level when creating or updating a Job, while meta-parameters to be updated before running are created and adjusted at an Instance level.

Results: Model results are returned for any meta-parameter being recorded at a job level. The results are returned as a JSON representation of the results class which can be parsed and used as needed. Results meta-parameters need to be formatted in a URL compliant way, e.g. "Confluence\\Outlet Node1\\Downstream Flow Volume" becomes "Confluence%5COutlet%20Node1%5CDownstream%20Flow%20Volume".

Parameter adjustment: Parameters are adjusted using meta-parameters. These meta-parameter relationships are created and shipped as part of a Source project through the Source GUI. Each Source Instance has a string array[] of meta-parameters to be adjusted at runtime. The format is "name=$SnowmeltGR4JRainfallRunoffModel_x1&value=100" and they can be updated through the PUT api/Instance?jobId=jobId call.

RIVER MODELLING UNCERTAINTY

Uncertainty analysis and calibration

The natural world is highly variable. Historical observations can give us an indication of future trends, but can also bias the interpretation of predictions through the believed existence of an expected outcome. While modelling ideally should give more accurate predictions deterministically through the capturing and simulation of physical processes, models are often forced by data produced from the analysis of observed trends. The combination of processes that cannot be fully expressed in models, with forcing data representing a single possible future, leads to a large range of potential solutions for any model predictions. The uncertainty around this range of potential model predictions needs to be quantified and captured so that it can be expressed as important context for the interpretation of modelled results.

Naively the best way to capture uncertainty is to vary all model parameters that are not known or measured through the full range of physically feasible values, combined with an ensemble of possible scenarios for the forcing inputs to the model. To properly express this in anything more than a trivial mode would require many tens of thousands, or even millions of model runs in a controlled environment. This is known as a Monte Carlo simulation (von Neumann and Ulam 1951 [8]). Fortunately there are methods to narrow down the potential parameter ranges to make the problem more tractable, but they still require a very large number for model simulation runs.

In a similar way, a calibration analysis uses a large number of model runs to search through parameter space for a set of model parameters that best match model outputs to observation data not used in the model itself. Both model calibration and uncertainty analysis are computationally intensive, which requires modellers to have access to expensive computation infrastructure. As calibration and uncertainty analysis are more routinely undertaken in operational agencies, access to this large scale computational infrastructure can be limited; this is especially the case in the developing world. The organisationally agnostic nature of cloud computing combined with its virtually limitless potential to scale makes it a very attractive resource for uncertainty analysis and model calibration.

eWater Source

The product used in this case study is the hydrological modelling tool eWater Source. Source represents the physical and managements aspects of water movement and storage within a catchment, including runoff generation in catchments, storage in reservoirs and irrigation demand modelling. Source includes functionality for modelling the complex management rules associated with regulation in Australian rivers (Penton and Gilmore 2009 [5]). This functionality aims to replicate that of ageing Australian models used to underpin water sharing plans such as Integrated Quantity Quality Model (Simons and Podger 1996 [7]), Resource Allocation Model (Perera, James et al. 2005 [6]) and Monthly Simulation Model-Bigmod (Close 1996 [2]).

The hydrological model used in the case study is a small catchment in Nepal with rainfall runoff derived through a modified GR4J algorithm. The simulation runtime of the model is around 4 seconds.

Uncertainty using DREAM

Estimation of hydrological models with non-physical parameters is typically automated through efficient search algorithms such as Shuffled Complex Evolution. Where multiple parameter values provide similar performance scores (nearly always), practitioners can apply informal uncertainty algorithms (e.g. GLUE) or formal uncertainty algorithms (e.g. Bayesian Monte Carlo Markov Chain) to provide guidance on the variance of particular parameters. In this case study we use the tool DREAM (Vrugt, ter Braak et al. 2009 [9]) which contains a formal MCMC approach using the R-based interface described by Joseph and Guillaume [3] (2013).

DISCUSSION

The results from the testing of the Source based Modelling Service Interface are still very preliminary. A small DREAM based analysis was setup by first adjusting DREAM to use RESTful calls when talking to its worker nodes. An analysis was then performed first with DREAM, SourceAdapterAPI and SourceBroker all running on the host machine, and then with SourceBroker on a remote machine on the same local area network. The results were very similar (Table 1). It is interesting to note that the results become faster the more evaluations take place suggesting some sort of initialisation overhead, or perhaps increased efficiency for multi-runs. This has yet to be investigated by the authors.

Table 1: Timings from DREAM test uncertainty analysis in seconds. On Windows 7 x64, 24GB of RAM, Intel x5650 CPU at 2.67GHz with 6 cores.

Number of Nodes	Function Evaluations	Total Time	Normalised Time	Send	Receive	Node Average Time
4	400	711.9	1.77975	0.19	0.1	593.7775
8	800	985.3	1.231625	0.45	0.16	779.98375
16	1600	1678.15	1.04884375	0.97	4.53	1192.27125

For comparison, using the same computer and DREAM analysis problem, but using a command line version of Source, each evaluation was around 2.4 seconds as opposed to the MSI version shown as normalised Time above. This is encouraging as the web based MSI version is often twice as fast.

CONCLUSIONS

The very preliminary results presented in this paper have shown that a web based modelling system for distributing complex modelling software to cloud based infrastructure can be a viable alternative to the more traditionally used HPC clusters, allowing more flexibility and almost limitless scalability. Further investigation is needed around the impacts of more complex problems and how distributing the MSI system components affect the systems performance.

ACKNOWLEDGMENTS

This work was funded by CSIRO's Water for a Healthy Country Flagship's Integrated Water Resources Management Theme, through the Water Information Technologies Stream.

REFERENCES

[1] Carr, R. and G. D. Podger (2012). Australia's Next Generation IWRM Modelling Platform. 34th Hydrology and Water Resources Symposium (December 2012), ISBN 978-1-922107-62-6
.

[2] Close, A. (1996). A new daily model of flow and solute transport in the River Murray. Hydrology and Water Resources Symposium 1996: Water and the Environment; Preprints of Papers, Institution of Engineers, Australia.

[3] Joseph, J. F. and J. H. A. Guillaume (2013). "Using a parallelized MCMC algorithm in R to identify appropriate likelihood functions for SWAT." Environmental Modelling & Software 46: 292-298.

[4] Leighton, B. P., P. Manser, D. J. Penton, J. Shoesmith, M. P. Stenson and J. Vleeshouwer (2011). "Exposing a Hydrological Simulation Model on the Web." 19th International Congress on Modelling and Simulation (Modsim2011): 1230-1236.

[5] Penton, D. J. and R. Gilmore (2009). Comparing software for modelling the management rules that river operators implement. 18th World IMACS / MODSIM Congress, Cairns, Australia, 13-17 July, 2009.

[6] Perera, B. J. C., B. James and M. D. U. Kularathna (2005). "Computer software tool REALM for sustainable water allocation and management." Journal of Environmental Management 77(4): 291-300.

[7] Simons, M. and G. D. Podger (1996). "IQQM – A hydrologic modelling tool for water resource and salinity management. ." Environmental Modelling and Software 11: 185-192.

[8] von Neumann, J. and S. Ulam (1951). "Monte Carlo Method." National Bureau of Standards Applied Mathematics Series 12: 36.

[9] Vrugt, J. A., C. J. F. ter Braak, H. V. Gupta and B. A. Robinson (2009). "Equifinality of formal (DREAM) and informal (GLUE) Bayesian approaches in hydrologic modeling?" Stochastic Environmental Research and Risk Assessment 23(7): 1011-1026.

[10] Welsh, W. D., J. Vaze, D. Dutta, D. Rassam, J. M. Rahman, I. D. Jolly, P. Wallbrink, G. M. Podger, M. Bethune, M. J. Hardy, J. Teng and J. Lerat (2013). "An integrated modelling framework for regulated river systems." Environmental Modelling & Software 39: 81-102.