sdCloud:
Cloud-based computation environment for System Dynamics models

Ivan A. Perl, PhD.	Robert Ward
Saint Petersburg (Russia) National Research University of Information Technologies, Mechanics and Optics (ITMO) ivan.perl@outlook.com	Iowa State University 785-424-3856 robertw@iastate.edu

Abstract

This paper describes a new open source project (sdCloud) dedicated to creating a cloud-based execution environment for system dynamics models. The goal is to provide model sharing and remote model execution and result generation. This solution is targeted to become publicly available at the end of June 2016.

Keywords: XMILE, model exchange, model execution, PySD model export, sdXchange model translation, System Dynamics

Introduction

Personal, portable, and mobile computers have become very powerful, but even so don't always have enough computational capacity for all simulation applications. Especially for Big Data applications, researchers often need faster execution and high bandwidth connections between the execution engine and data, regardless of where the data resides. It is exactly this kind of need that has given rise to cloud computing and cloud storage paradigms.

Cloud-based execution is particularly convenient for Big Data applications and research, as keeping both the data and the execution engine in the cloud reduces the modeler's involvement in data sharing and resource management. Cloud-based engines are also an archetypical match for XMILE encoded models, allowing researchers to select the execution engine somewhat independently of the model creation tool. Especially in certain research domains, matching the engine technology to the data processing context has

significant benefits. Further, additional synergies are possible when XMILE translation resources are also available in the cloud. Cloud-based translators are more accessible to typical users than translators which require local installation. Locating both translators and engines in the cloud also simplifies translator validation and testing. This paper describes our general approach to cloud-based translation and model execution based on two specialized engines: PySD, which is almost in production-ready state, and ErlSD, which is currently under development.

Motivation

Superficially, the technological layer of system dynamics modelling appears feature rich and mature. Commercial products (like those from ISEE, Ventana, AnyLogic, Simulistics, and others) provide rich modelling functionality covering most of the common use cases. Forio-Simulate features a cloud-based model execution and presentation environment for selected proprietary modelling languages.

There are also open source and free projects like Minsky, Mapsim, and Simantics SysDyn. Paralleling the migration of applications to web-based deployments in other domains, web-based modelling approaches have emerged, offering a kind of fusion modelling approach. InsightMaker and Systo, for example, allow model development in almost any Web browser supporting JavaScript.

Despite this highly evolved modelling ecosystem, there are some limitations. In particular, until the formalization of XMILE [1], it was difficult to separate model creation and model execution [2][3]. This separation is particularly important in Big Data work, because the data itself is often coupled to a particular storage and processing technology. Thus the tool which might be most effective for developing a particular model is often not the most effective tool for executing the model on the relevant data store.

In discussions stimulated by last year's conference between sdXchange [4], PySD [5] and others interested in model exchange, it became clear that a cloud-based deployment would be the most effective way to address the Big Data needs, and that cloud-based model translation, execution, and sharing would not only create synergies with existing system dynamics tools, but also be a hospitable and open house for future development and research.

Thus, just as many different services that we once hosted on our local machines have moved to the cloud -- for example, habitual database servers migrated to the cloud and became DBaaS (Database as a Service); various software platforms became PaaS (Platform as a Service) - Project sdCloud is an attempt to move model translation, computation, and sharing to the cloud. Think of sdCloud as MaaS (Modelling as a Service.)

The sdCloud

As with other cloud-based solutions, an important benefit of sdCloud is that the cloud service not only executes the model, but also manages all resource-related issues and is easily accessible from almost everywhere via almost any connected device.

While the key goal of this project is to build a cloud-based system, running on powerful, well-connected servers, other requirements, though, seemed necessary to reap the full benefit from a cloud modelling service. Thus, we expect sdCloud to provide following set of features to the end-user:

1. The ability to upload a system dynamics model in any format, including XMILE and those formats supported by sdXchange.
2. The ability to download a previously uploaded model in any format supported by sdXchange, independently from the format in which it was uploaded.
3. The ability to schedule model execution, track schedule and execution progress, and browse model execution results.
4. The ability to share models and model execution results both within sdCloud and outside it.

The sdCloud project does not aim to build a complete solution from scratch. Instead, we hope to build a management and access layer on top of components created by others. Our initial work will draw on components from three key community-supported projects: sdXchange, PySD, and ErlSD.

SdXchange: Model Translation

We will rely upon translators from the sdXchange project. Introduced at the 2015 System Dynamics conference the key goal of sdXchange is to create modular standalone translators for SD models [1]. These translators will allow sdCloud to upload models in one format (e.g., including orphaned languages like DYNAMO) and convert them to another modelling system (e.g., Stella). By having sdXchange components inside of the cloud, we will allow end users to upload for further processing not only XMILE models, but also models, which can be translated by sdXchange.

Furthermore, because, for any model type supported by sdXchange there is a translation into and from XMILE format, using these translators, sdCloud can exploit not only XMILE-aware execution engines, but also other engines. By building a translation pipeline using XMILE as an intermediate form, the sdCloud front end can translate from the uploaded model format to the format of the desired engine for any transitive combination supported by sdXchange. For example, to execute a model captured in one format on an engine designed for a second format (not XMILE) sdCloud could compose the input pipe modelFormatA -> XMILE -> modelFormatB.

The relationship between sdCloud and sdXchange is fully symbiotic: sdXchange will supply flexibility and translation expertise to sdCloud; sdCloud will develop a friendly, web-based user interface for sdXchange translation (even to users not interested in cloud execution).

Of course, not all models will need to be translated; those users fortunate enough to have a modelling environment capable of saving in XMILE will be able to upload their models and execute directly from XMILE.

Model Computation

Just as sdCloud will delegate model normalization to modular translators, it will also delegate module execution to modular "computation cores". This strategy not only avoids re-implementing complex execution engines, it also facilitates continuous integration of new capabilities. Development of model execution systems is currently a rich area for engineers and researchers, especially for those focused on highly efficient computation, parallelization, and big data. Since both modelling technologies and cloud environments are evolving, it is important that the sdCloud solution avoid dependencies on specific cloud or execution implementations. Treating the execution engine as a "pluggable" extension point, allows us to quickly integrate improvements in sdCloud-supplied services with both existing and new model execution engines.

Our initial implementation will delegate to an engine provided by the PySD project [3]. We are actively developing a second engine ErlSD. Later, when ErlSD and other new options become available and are added to the cloud deployment, end-users will be able to choose which core to use in each particular model execution scheduling.

PySD computation core

This project is a library for running System Dynamics models in python, with the purpose of improving integration of Big Data and Machine Learning into the SD workflow. PySD translates Vensim model files into python modules, and provides methods to modify, simulate, and observe those translated models. Executable models created using PySD are represented by Python modules and they can be used for models analysis, development and debugging. Since the generated Python code is quite readable, sdCloud will not only use it as input to the computation core, but will also allow downloading of translated python models.

ErlSD computation core

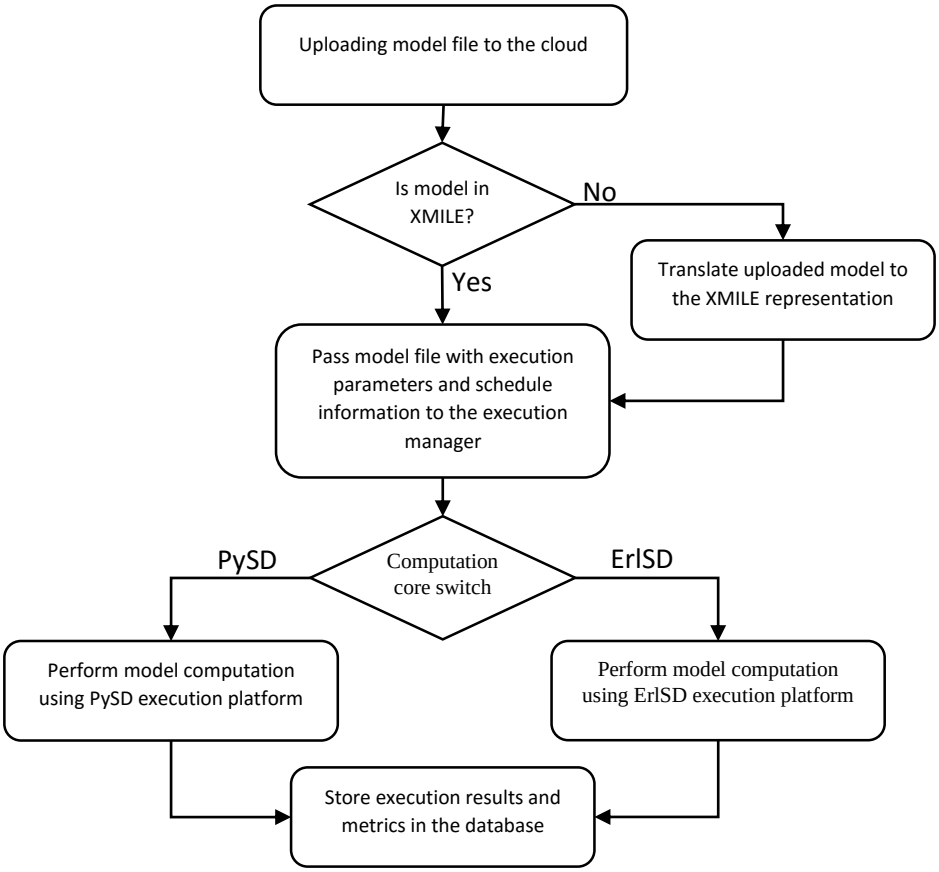
ErlSD is being developed as a second execution engine for the cloud deployment. Like PySD, ErlSD will convert a system dynamics model into an executable module. However, unlike PySD, the ErlSD execution module will be written in Erlang. While Python has the advantage of being familiar to researchers in many areas, allowing them to directly modify the generated code if necessary, Erlang promises efficiency benefits for computationally intensive simulations. Models expressed in Erlang should be able to easily

exploit Erlang's advanced optimization and parallelization features. We also expect that models expressed in Erlang will be more naturally able to exploit the cloud's flexible resources allocation.

The sdCloud Solution

The sdCloud project will develop the user interface, resource management, and integration glue needed to needed to compose a meaningful cloud solution from sdXchange, PySD, and ErlSD components (and future extensions.) Independently of computation/execution approach used in the system, it will need to take an XMILE model as an input, perform modeling and produce model execution results. Some of the execution parameters (like computation time, CPU and memory usage) will be recorded and tracked in a unified way. Since execution results be produced by different systems in different formats, cloud integration wrappers will be responsible for collecting and properly storing executor outputs in the database.

The following chart depicts the main flow for model execution.



An important choice when designing a cloud-based solution the resource scaling strategy. There are two major options: adjusting the number of virtual machines in the cloud and dynamically allocating additional physical hardware. A VM-based approach is appropriate when large numbers of end-users are constantly pushing data to the cloud. In the case of sdCloud, however, the most significant portion of the service load will come not from outside, but will be generated internally during model computation. That is why scaling issue is solved via dynamic allocation of model translation and execution nodes.

The sdCloud solution will be deployed in data centers of Saint-Petersburg National Research University ITMO, where it will use not only resources dedicated to this project, but will also be able access shared resources to dynamically adjust translation and computation capacities. Thus, sdCloud will use a resource scaling approach similar to what is common in many continuous integration environments (a 'la Jenkins [6]) When additional resources are required, a core node can deploy computation agents to new nodes and transfer part of the scheduled execution work there. This approach distinguishes between a master node (primarily responsible for job initiation and resource management) and a variable number of slave nodes where individual jobs can be transferred for execution.

For early access, the deployed sdCloud solution will be available in test mode at http://sdCloud.ifmo.ru (starting at beginning of May.) The first general release is scheduled for deployment at the end of June.

Conclusion

The sdCloud project is an answer to the big data processing and data access challenges facing the System Dynamics community. A high performance cloud-based environment can make working with system dynamics easier, faster and closer to the model's users. Naturally, a cloud-based solution reduces the need for users to manage local installation of various tools. However, sdCloud has potential benefits beyond its utility to those who would like to execute, translate or share their models. This is also an open platform where new tools can be deployed, thus, it should attract not only modelers, but also tool developers and thus stimulate further innovation in System Dynamics.

References

1. OASIS XML Interchange Language (XMILE) for System Dynamics TC (https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xmile)
2. Chichakly, K. 2007. SMILE and XMILE: a common language and interchange format for system dynamics. Proceedings of the 2007 International System Dynamics Conference, Boston, MA. System Dynamics Society: Albany, NY.
3. Diker, V.G. and Allen, R.B. 2005. XMILE: towards an XML interchange language for system dynamics models. System Dynamics Review 21(4)
4. Ward, R., Houghton, J., and Perl, I. A. (2015) SDXchange: stand-alone translators to enable XMILE model adaptation, transportation, and exchange. Syst. Dyn. Rev., 31: 86–95. doi: 10.1002/sdr.1529

5. Houghton J., Siegel M. (2015) Advanced data analytics for system dynamics models using PySD. Proceedings of the 33rd International Conference of the System Dynamics Society
6. Jenkins continuous integration system https://jenkins-ci.org/