Advances in Engineering Software xxx (xxxx) xxx–xxx

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

Research paper

Engineering design analysis utilizing a cloud platform

Sunil Suram, Nordica A. MacCarty, Kenneth M. Bryden*

Department of Mechanical Engineering, Iowa State University, 1620 Howe Hall, Ames, IA, 50011, United States

ABSTRACT

In this paper we present a novel methodology for modeling engineered and other systems based on integrating a set of component models that are accessible as "model-as-a-service" components within a cloud platform. These component models can be combined together to form a systems model. The component models are stateless and web-enabled. The advantage of being web-enabled is that developers can use the models as API endpoints as opposed to library components, hence making the models themselves language agnostic and less restrictive in their use. These ideas are presented within the context of a previously published engineering model for the thermal analysis and preliminary design of a small biomass cookstove. In this paper the monolithic biomass cookstove model is separated into six independent, stateless component models supported by a generic model application infrastructure. Interaction between the models is orchestrated by a federated model system. Finally, the results of the cookstove from the monolithic model were compared with the distributed systems model. It was found that there was no change in the results. However, the systems model increased the time-to-solution due to network latency. However, the ability to share models and data via API endpoints, will likely offset the overall wall-clock time for model integration, since model developers do not have to make code changes. In conclusion, it is advantageous to build web-enabled component models for their easy reuse across multiple systems models.

1. Introduction

Engineered systems are generally composed from interdependent components that are themselves composed of other subcomponents. Because of this, the representation and analysis of the detailed interactions within and between the components comprising these engineered systems is critical. However, holistic modeling of these systems is challenging [2]. This is in part due to the size and complexity of detailed systems models and in larger part because of the need to organize and integrate a collection of models representing each of the components and subcomponents of the system into a single coherent information artifact. This is particularly challenging because in many cases each of the component and subcomponent models are developed by separate teams of analysts (or individual analysts) with differing domain knowledge, differing modeling practices, and differing expectations as to the outcomes of the modeling and analysis process.

Integrated modeling [13,19] seeks to address these challenges and "includes a set of interdependent science-based components (models, data, and assessment methods) that together form the basis for constructing an appropriate modeling system" [13]. In a traditional modeling approach, the individual component models are linked with each other using software/code. In many scientific and engineering applications each of the submodels is incorporated as a subroutine within the larger systems code. In other cases each of the models is a software library that exposes application programming interfaces (APIs) [17]. All the linking occurs in a single software program that brings together the individual models, including problem specific entities like initial conditions, boundary conditions and information/data transfer between models, all of which must be mediated and controlled by the applications developer.

One way of overcoming the problem of building and modeling large-scale systems models is by taking an *integration framework* based approach, where each individual model is part of a larger framework of models [23]. There are a wide range of integration frameworks which are based on coupling the models together by linking the inputs and the outputs together in the manner of message passing between the component packages (i.e., models) in a way that requires adherence to a given data standard or requires user intervention to identify and manage the data flow. The environmental modeling community has been active in the development of a number of general purpose model integration tools [13]. All of these systems require models to have *initialize, run, finalize, get,* and *set* functions for basic control over the models; provide a code-based method for connecting the models together, using some form of XML file or some other type of configuration

* Corresponding author.
E-mail address: kmbryden@iastate.edu (K.M. Bryden).

http://dx.doi.org/10.1016/j.advengsoft.2017.10.004
Received 31 March 2017; Received in revised form 19 September 2017; Accepted 18 October 2017

file, to create a low-level model-to-model interface; and require an agreed upon global ontology describing the variables passed between models. Nearly all of these general purpose integration frameworks require that the models in the system use the same programming language.

One approach to overcoming the challenges of model integration frameworks is to develop loosely coupled and decentralized systems similar to a web-based application that, for example, predicts turnout at a public event based on location, weather, and traffic conditions. Each of the services (public event listing, weather, and traffic conditions) is an independent information service accessed over the Internet. Each of these services is developed independently of the others without making any assumptions or following common data interchange protocols. Only the service that combines the data from each of the three information services needs to know the protocol emitted by each service it is accessing. At a future date if the wrapper service needs to add, say, public transportation information, it can do so by calling yet another service that publishes this new piece of information. This can be done without renegotiating previously established protocols. Such a decentralized approach leads to a service-oriented architecture with cleaner interfaces and data transfer between each service [22].

In this article, a novel approach to developing a loosely coupled and decentralized integrated modeling environment appropriate for engineering and scientific computing is proposed, which is based on a federation of independent models each of which is an independent web-based model service (i.e., an information artifact) accessible via a web API using interaction protocols chosen by the model developer. In the proposed framework, this collection of independent models is joined together in a federation which describes a particular system. In this way, a complex system model can be built by bringing together multiple individual models and can be deployed as a system model. Furthermore, the developed models within the federation do not have any schema imposed on the structure of inputs and outputs by the developer. The model developer decides the structure of inputs and outputs that the model accepts and emits, the only requirement being that this information be broadcast to model developers via API calls. In this way a component model can be used as-is in multiple systems models. The communication between constituent models is orchestrated by a federation management system (FMS).

For such a decentralized system of models to be functional and scalable a cloud-based architecture is the most practical solution. The primary reasons for this are

- Universal availability—Cloud platforms can be accessed by anyone with a web browser and an Internet connection. This opens up the possibility for model developers to build and publish their models either as part of a closed group or globally regardless of their geographic location.
- Scalable platforms—As more models are added the computational resources can be increased automatically without human intervention. The deployed models need to be readily available every time the FMS sends a request for computation.
- Cost effective—Cloud platforms work on a cost per usage model and are hence cost effective because the user only pays for the compute time and not for procuring, provisioning, and maintaining the compute, storage, and networking resources.

2. Background

2.1. Cloud computing

Over the last few years with the rise of cloud computing, it has become easier to obtain compute cycles on an on-demand basis [1]. Several companies have built data centers and technology platforms using commercial off-the-shelf hardware and are making them available over the Internet. This has enabled applications to scale dynamically to support millions of concurrent users for consumer applications. The same technology platforms are now being used in traditional enterprise applications, blurring the difference between consumer and enterprise applications [8]. Software vendors now host thousands of applications in the cloud that can be accessed by its users through a web-browser. Cloud computing eliminates the purchase of expensive processors, networking, and storage resources by making them available over the Internet [1]. The availability of high-end hardware on an on-demand basis makes it possible to move many types of engineering, scientific, and HPC applications and workflows into the cloud. This has opened up new opportunities for creating novel scientific, data management, analysis, and visualization applications that leverage the performance capabilities offered by cloud computing. Engineering modeling and scientific computing problems can now be solved by leveraging cloud-computing capabilities. By harnessing the computing power of the cloud, smaller form-factor devices can also be integrated into engineering workflows and perform operational tasks in the field that might not be currently possible. It also becomes possible to scale applications at a lower cost per processing/computing unit [3].

Several researchers ([12,21,28], as well as commercial entities [1,18] have made attempts to run HPC workloads on cloud computing resources. Data from astronomical measurements was processed utilizing cloud computing resources from Amazon, Nimbus, and Eucalyptus by provisioning cloud computing resources, mapping workloads to them, and de-provisioning the resources on completion [28]. They found that cloud computing can be a viable solution for several scientific computing problems. As part of their research they also concluded “*being able to add and remove resources at runtime outweighs the networking and system management overheads.*” A platform for scientific computing was developed and used for simulations in materials science [12] using the Amazon Elastic Compute Cloud (EC2) [1] to develop an Amazon Machine Image that was the primary underlying technology for their platform. The authors solved two problems in materials simulations that involved loose and tight coupling of codes. They found that although the Amazon EC2 platform is not efficient for the transfer of large amounts of data, it is competitive in achieving the speedups similar to that provided by Infiniband clusters. A big data platform for scientific workflows was developed and used to solve an image processing problem [29]. They also compared the efficacy of multiple cloud platforms for performance, price, and ease of provisioning and management of compute resources.

A series of experiments were developed to evaluate the Amazon EC2 infrastructure as an alternative for many-task computing workloads [10]. Although the authors found that the overall performance of commercial cloud hardware is low compared to dedicated HPC resources, they underscore the fact that commercial clouds can fill the gap for temporary and instant need for compute resources. HPC clusters were extended using EC2-based cloud clusters by Belgacem and Chopard [4]. They found that with a load-balancing strategy they were able to benefit from utilizing the cloud computing resources, as opposed to merely adding more machines to an existing cluster. The authors used MPI-based models that were coupled and executed in a distributed manner.

The researchers above primarily have used cloud computing instances to extend existing HPC hardware and augment the time-to-solution for their specific scientific problems. Commercial cloud hardware was found lacking in some instances [28], but several researchers have acknowledged the power of being able to quickly provision and utilize on-demand compute resources [4]. Also, most of the computational workloads have been using traditional models like MPI, but running on cloud hardware. So most of the problems solved in the literature are related to comparing performance of existing codes running in the cloud.

2.2. Stateless models

The concept of state is critical to the implementation of the federated model sets described here. State refers to the entirety of