Efficient data access for Open Modeling Interface (OpenMI) components

Tom Bulatewicz, Daniel Andresen

Efficient data access for Open Modeling Interface (OpenMI) components

Tom Bulatewicz, Daniel Andresen
Kansas State University, Dept. of Computing and Information Sciences
234 Nichols Hall, Manhattan KS, 66506, USA
{tombz, dan}@ksu.edu

Introduction

Data management can be a challenging task when employing linked (or coupled) simulation models that execute independently and cooperate to collectively carry out a simulation. In the case of models that are software components with well-defined input/output interfaces, data access can be simplified by linking general-purpose data components to model components. An interdisciplinary team of engineers and scientists at Kansas State University are integrating crop, hydrological, and economic models toward developing a comprehensive understanding of agricultural systems (NSF grant GEO0909515). These multidisciplinary models are linked together using the Open Modeling Interface (OpenMI) which defines a standard way for software components to exchange data with each other and coordinate their execution. In this work we present the design of a general-purpose Data Provider Component (DPC) that is capable of delivering data from online sources to OpenMI components.

Methods

Fig. 1 illustrates the movement of data through a distributed data delivery system for linked model components. Compositions of linked components execute on cluster nodes. Each composition includes a DPC that retrieves data from web services and provides it to the other components. DPCs within compositions that are running on different cluster nodes share data with each other. When a model component needs input data it invokes the *getvalues* function of the DPC for the needed quantity, time, and locations (*element set*) and the DPC returns the appropriate set of values (a *valueset*). The DPC calls web services for a specific quantity identifier, time, and list of location identifiers and then extracts the values from the response. Any web service that can be queried for a quantity, time, and list of locations and returns a list of values could be used as a data source for a DPC. The current implementation supports WaterOneFlow web services (e.g. DAYMET) that provide time series data in XML that conforms to the WaterML schema.

To minimize the impact that the DPC has on the execution time and resource use of a linked simulation, it must (1) minimize the *wait time*, which is how long it takes for the getvalues function to return a valueset and (2) minimize the number of times each valueset is retrieved from a web service. To these ends, the DPC utilizes three strategies: caching, prefetching, and pipelining. Each valueset is retrieved from web services once and is then cached so that it is immediately available for subsequent requests by other model components (within and across compositions) and subsequent executions of the composition. Since components typically advance forward through simulation time, valuesets can be prefetched so that they are available in the cache before they are requested. Multiple web service calls are performed simultaneously in a pipelined fashion to maximize use of available network bandwidth.

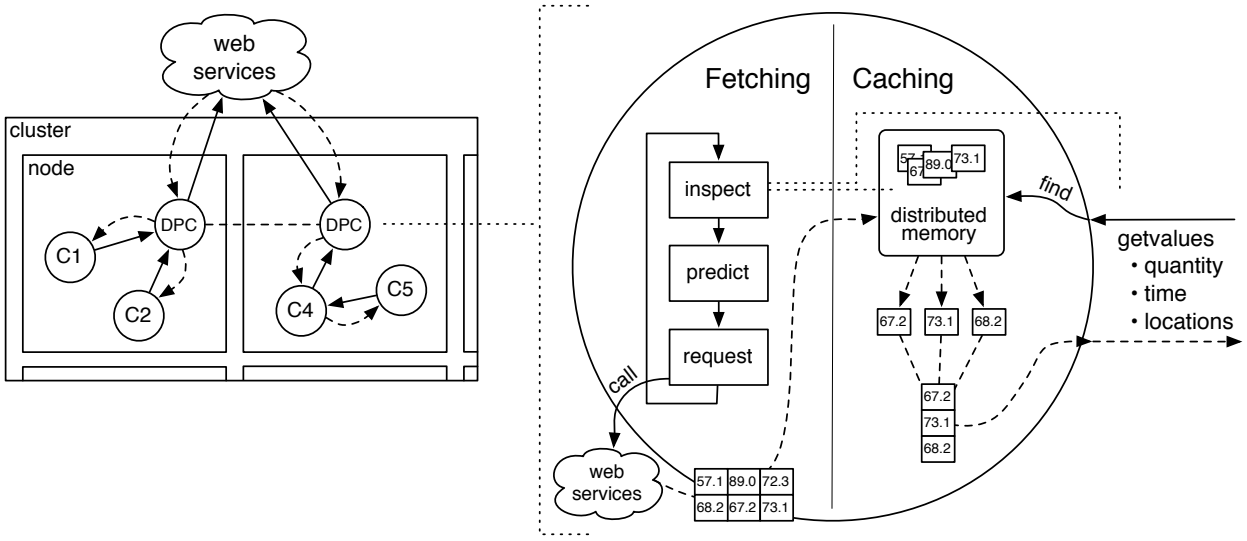

Figure 1: System overview (left) and operation of the data provider component (right). Solid lines indicate requests and dashed lines indicate the flow of data.

The DPC consists of a fetching module and a caching module (Fig. 1 right). The fetching module continuously identifies the valuesets that the model components have requested or are expected to request, retrieves them from the web services, and then stores them in memory. The caching module waits for requests from model components and responds by retrieving the appropriate values from the cache, assembling the valueset, and returning it to the calling model component. The fetching module predicts which valuesets the model components may request based on their previous requests. Valuesets are prefetched to the same point in simulation time for all model components linked to a DPC and prefetching is performed only when there are available system resources (based on CPU and network usage).

Results and discussion

We implemented the DPC and evaluated its performance and efficiency. Caching within a linked model resulted in constant wait time as the number of model components increased. Distributed caching resulted in reduced wait time as a function of the number of concurrent simulations and the time required to retrieve data from the web services. In both cases, the amount of data transferred was minimized and remained constant as the number of model components increased. When a sufficient number of concurrent requests are permitted, prefetching and pipelining resulted in constant wait time as the number of model components increased. To mitigate some of the challenges of data management for linked simulations, intelligent, efficient data provider components will become an essential part of any OpenMI linked model. We believe that this work provides a sound basis for the development of such components.

References

J. B. Gregersen, P. J. A. Gijsbers, and S. J. P. Westen. OpenMI: Open modeling interface. J. Hydroinform., 9(3):175–191, 2007.