Decentralized Orchestration of Data-centric Workflows in Cloud Environments

Bahman Javadi[a,*], Martin Tomko[b], Richard Sinnott[c]

[a] *School of Computing, Engineering and Mathematics*
University of Western Sydney, Australia
[b] *Faculty of Architecture, Building and Planning*
The University of Melbourne, Australia
[c] *Department of Computing and Information Systems*
The University of Melbourne, Australia

Abstract

Data-centric and service-oriented workflows are commonly used in scientific research to enable the composition and execution of complex analysis on distributed resources. Although there are a plethora of orchestration frameworks to implement workflows, most of them are unsuitable for executing (enacting) data-centric workflows since they are based on a centralized orchestration engine which can be a bottleneck when handling large data volumes. In this paper, we propose a flexible and lightweight workflow framework based on the Object Modeling Systems (OMS). Moreover, we take advantage of the OMS architecture to deploy and execute data-centric workflows in a decentralized manner across multiple distinct Cloud resources, avoiding limitations of all data passing through a centralized engine. The proposed framework is implemented in the context of the Australian Urban Research Infrastructure Network (AURIN) project which is an initiative aiming to develop an e-Infrastructure supporting research in the urban and built environment domains. Performance evaluation results using spatial data-centric workflows show that we can reduce 20% of the workflows execution time when using Cloud resources in the same network domain.

Keywords: Data-centric Workflows, Cloud Computing; Orchestration, Object Modeling System;

*Corresponding author. Telephone: +61-2-9685 9181; Fax: +61-2-9685 9245
Email address: b.javadi@uws.edu.au (Bahman Javadi)

1. Introduction

Service-oriented architectures based on Web services are common architectural paradigms for developing software systems from loosely coupled distributed services. In order to coordinate a collection of services in this architecture to achieve a complex analysis, workflow technologies are frequently used. A workflow can be considered as a template to define the sequence of computational and/or data processing tasks needed to manage a business, engineering or scientific process. Although the workflow concept was originally introduced for automation of business processes, there is a significant interest from scientists to utilize these technologies to automate complex, often distributed experiments.

Two popular architectural approaches to implement workflows are *service orchestration* and *service choreography* [1]. In service orchestration, there is a centralized engine that controls the whole process including control flow as well as data flow. An example of this implementation is the *Business Process Execution Language* (BPEL), which is the current *defacto* standard for orchestrating Web services [2]. On the other hand, service choreography refers to a collaborative process between a group of services to achieve a common goal without a centralized controller. The *Web Services Choreography Description Language* (WS-CDL) is an example of this type of implementation based on the XML language [3].

The main issue with service orchestration implementations is transferring all data through a centralized orchestration engine, which can be a bottleneck for the performance, especially for data-centric workflows. To tackle this problem, we introduce a new framework to implement data-centric workflows based on the Object Modeling System (OMS). OMS is a component-based modeling framework that utilizes an open-source software approach to enable users to design, develop, and evaluate loosely coupled cooperating service models [4]. The framework provides an efficient and flexible way to create and evaluate workflow models in a scalable manner with a good degree of transparency for model developers[1].

The OMS framework is currently being used to design and implement a range of scientific models [6]. However, the support of this framework for

[1]This paper is an extended version of [5].

data-centric and service-oriented workflows has not been investigated, which is the main goal of this paper. Although the OMS framework can be generally classified as a service orchestration model, we show how we can take advantage of the OMS architecture to implement a decentralized service orchestration overcoming the limitations of centralized data flow enactments. This feature is crucial for data-centric workflows that deal with large quantities of data and data movement in situations where the use of a centralized engine could decrease the performance of the workflow or indeed make certain workflows impossible to enact.

The proposed framework is implemented in the context of the Australian Urban Research Infrastructure Network (AURIN)[2] project, which is an initiative aiming to develop an e-Infrastructure supporting research in the urban and built environment research disciplines [7] as outlined in Figure 1. It will deliver a *lab in a browser* infrastructure providing federated access to heterogeneous data sources and facilitate data analysis and visualization in a collaborative environment to support multiple urban research activities.

We evaluate the proposed architecture through enactment of realistic data-centric workflows exploiting data gathered from federated services communicating based on standardised protocols as defined by the Open Geospatial Consortium (OGC)[3]. These data are then used for the computation of topology graphs, a basic building block for many spatial analytical tasks. The performance evaluation experiments have been conducted on different Cloud infrastructures to assess the flexibility and scalability of the proposed architecture. In summary, the contribution of this paper is as follows:

- We propose a flexible workflow environment for scientific workflows based on the Object Modeling System;
- We propose a decentralized orchestration model to reduce the execution of data-centric-workflows;
- We implement the proposed workflow environment in the context of the AURIN project;
- We evaluate the proposed architecture using realistic workflows in the urban research domain.

[2]http://aurin.org.au/
[3]http://www.opengeospatial.org/

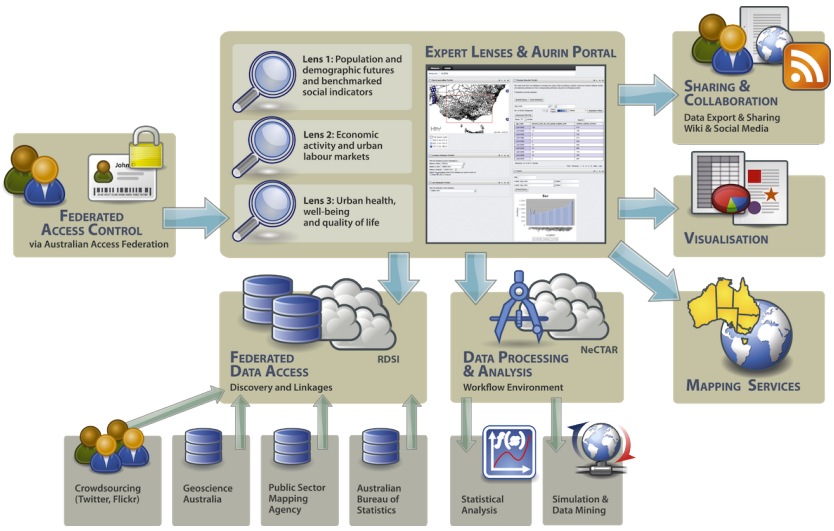


Figure 1: The overview of the AURIN conceptual architecture.

The rest of the paper is organized as follows. We provide an overview of the AURIN project in Section 2. In Section 3, we present the Object Modeling System framework. Section 4 explains the implementation of data-centric workflows using the OMS framework. The performance evaluation of the proposed architecture is presented in Section 5. In Section 6, a case study using the workflow environment in the AURIN project is presented. The related work is also illustrated in Section 7. We finally conclude our findings and discuss the future work in Section 8.

2. The Wide-area Context of AURIN

In this section, we provide an overview of the AURIN project and its operational context, which directly motivates the research presented in this paper.

2.1. AURIN System Overview

The AURIN project is tasked with developing an e-Infrastructure through which a wide range of urban and built environment research activities will be supported. The AURIN technical architecture approach is based on the

concept of a single sign-on point of entry portal[4] (Figure 1). The sign-on capability is implemented through the integration of the Australian Access Federation (AAF)[5], which provides the backbone for the Internet2 Shibboleth-enabled[6] decentralized identity provision (authentication) across the Australian university sector. The portal facilitates access to a diverse set of data interaction capabilities implemented as JSR-286 compliant portlets. These portlets represent the user interface component of the capabilities integrated within a loosely coupled service-oriented architecture, exposing data search and discovery, filtering and analytical capabilities, coupled with mapping services, and various visualization capabilities.

The federated datasets feeding into AURIN are typically accessed through programmatic APIs. A rich library of local (e.g., Java) and federated (REST or SOAP services) analytical tools is exposed through the workflow environment based on the OMS framework. The dominantly spatial nature of datasets used in the urban research domain requires the interfacing with services implementing OGC standards for access to federated data resources. In particular, the Web Feature Service (WFS) standard implementations [8] represent one of the most common sources of urban spatial data, served through spatial data infrastructures. Data acquired through such services are analysed by analytical processes for advanced statistical analysis of spatial and aspatial data, thus exposing a complex modeling environment for urban researchers.

The workflow environment presents an important backbone of the AURIN infrastructure, by supporting:

- complex data-centric workflows to be repeatedly executed, leading to a better reproducibility of data analysis and scientific results;
- workflows that can be re-executed with altered parameters, thus effectively supporting the generation of multiple version of scenarios;
- workflows that support the interruption of the analysis design process, enabling research spanning across extended periods of time;
- workflows that can be shared with collaborators and used outside of

[4]http://portal.aurin.org.au
[5]http://www.aaf.edu.au/
[6]http://shibboleth.internet2.edu/

AURIN;

- workflows that are encoded in a human-readable manner, effectively carrying metadata about the analytical process that can be scrutinized by peers, thus supporting greater transparency and research quality.

The results of data selection and analysis can be fed in to a variety of visual data analytics components, supporting visual exploration of spatio-temporal phenomena. 2D (and 3D soon) visualization of spatial data, their temporal filtering, and multidimensional data slicing and dicing are amongst the most sought-after components of AURIN, that will be integrated with a collaborative environment. This allows researchers from geographically remote locations to collaborate and coordinate their efforts on joint research problems.

AURIN is also leveraging the resources of other Australian-wide research e-Infrastructures such as the National eResearch Collaboration Tools and Resources (NeCTAR)[7] project, which provides infrastructure services for the research community, and the Research Data Storage Infrastructure (RDSI)[8] project, which provides large-scale data storage. At the moment, the AURIN portal is running on several virtual machines (VMs) within the NeCTAR NSP (National Servers Program) while we utilize both the NeCTAR Research Cloud and the Amazon's EC2 public Cloud as the *processing infrastructures* to execute complex workflows.

2.2. Wide-area Analytical Processing with Federated Data

At an abstract level, the required time to execute a given workflow W is based on the time for workflow composition and orchestration (T_{W_C}), the time to retrieve the data at the (federated) data source (T_{W_D}), the time for processing the data (T_{W_P}), and the time to transfer the data between the data source and the processing infrastructures (T_{W_X}). So, we have:

$$T_W = T_{W_C} + T_{W_P} + T_{W_D} + T_{W_X} \tag{1}$$

The aim of this research is to devise a workflow environment to minimize the workflow execution time (i.e., T_W). This time is the summation of the time intervals required to complete each of the aforementioned elements,

[7] http://nectar.org.au
[8] http://rdsi.uq.edu.au

which is directly proportional to the size of the messages passed, data processed, and the computational complexity of the processing algorithm. To do this, we focus on reducing the transfer time of the data to the processing infrastructures (i.e., T_{WX}) by decentralized orchestration and execution of the data-centric workflows.

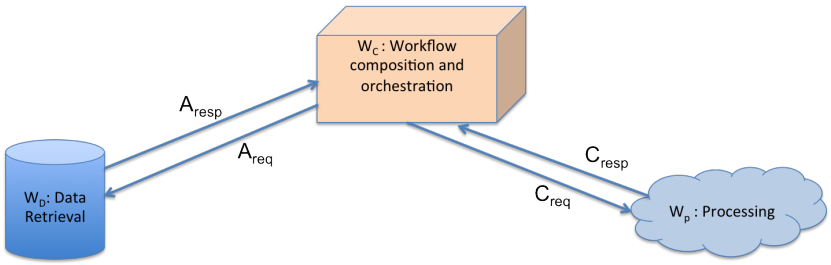


Figure 2: Centralized workflow execution with data acquired from a federated data source.

Consider a traditional centralized workflow execution as illustrated in Figure 2, where the data is acquired from a federated data source and passed thorough the workflow engine to be analyzed by the processing infrastructures. In this scenario, the transfer time required by the workflow is the sum of the messaging time for passing the requests and responses to the federated data resource (A_{req} and A_{resp}), and the messages passed between the workflow engine and the processing infrastructures (C_{req} and C_{resp}). So the transfer time can be obtained as follows:

$$T_{WXc} = T_{A_{req}} + T_{A_{resp}} + T_{C_{req}} + T_{C_{resp}} \tag{2}$$

Assume a second scenario as depicted in Figure 3, where the data acquired from a federated data source is directly transfered to the processing infrastructure and only the results are sent to the workflow engine. Thus, the time spent on the data transferring can be calculated as follows:

$$T_{WXd} = T_{A_{req}} + T_{A_{resp}} + T_{B_{req}} + T_{B_{resp}} \tag{3}$$

In this case, the time for transferring the data to the processing infrastructure will be decreased as we bypassed the workflow engine. In other words, $T_{B_{req}} + T_{B_{resp}}$ is much lower than $T_{C_{req}} + T_{C_{resp}}$ due to the reduced amount of transferred data. This advantage is further magnified when we have a large volume of raw data in the data-centric workflows where sending

them to where they are required can cause considerable delays. These situations are all frequently met in the context of AURIN with the nationwide federated data sources. To this end, we propose a framework to adopt the second scenario to decentralize the orchestration and execution of data-centric workflows.

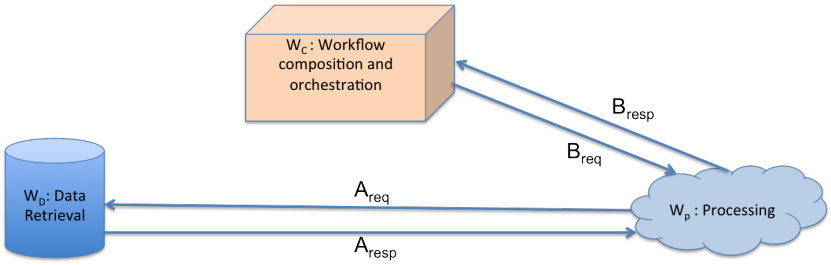


Figure 3: Decentralized workflow execution with data acquired from a federated data source.

3. Object Modeling System

The Object Modeling System (OMS) is a pure Java and object-oriented modeling framework that enables users to design, develop, and evaluate science models [4]. OMS version 3.0 (OMS3) provides a general-purpose framework to allow easier integration of such models in a transparent and scalable manner. OMS3 is a highly interoperable and lightweight modeling framework for component-based model and simulation development on different computing platforms. The term *component* is a concept in software engineering which extends the reusability of code from the source level to the binary executable. OMS3 simplifies the design and development of model components through programming language *annotations* which capture metadata to be used by the model.

The architecture of OMS3 is depicted in Figure 4. The core of OMS3 contains two main foundations: the knowledge base of the system (e.g., metadata and ontologies), and development tools and methods (e.g., science model). Other modules in this architecture include modeling resources and different modeling products. Interested readers can refer to [6, 4] for more information about the OMS3 architecture.

The main features of the OMS3 framework are:

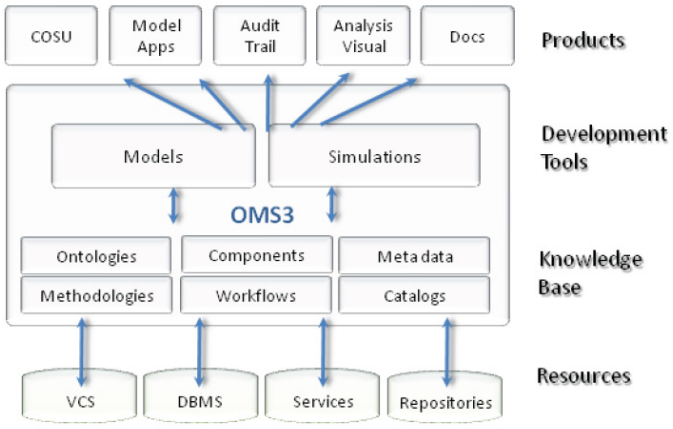


Figure 4: The Object Modeling System architecture [6].

- OMS3 adopts a *non-invasive* approach for model or component integration based on annotating 'existing' languages. Existing legacy classes are allowed to keep their identity, which means that once a component has been introduces into OMS3 it is still usable outside of OMS3. In other words, using and learning new data types and traditional application programming interfaces (API) for model coupling is mostly eliminated.

- The framework utilizes *multi-threading* as the default execution model for defined components. Moreover, component-based parallelism is handled by synchronizations on objects passed to and from components. The execution of components is driven by data flow dependencies. Therefore, without explicit programming by the developer, the framework is able to be deployed on multi-core Cluster and Cloud computing environments.

- OMS3 simplifies the complex structure for model development by leveraging recent advantages in *Domain Specific Languages* (DSL) provided by the Groovy programming language. In fact, a DSL is a language extension dealing with a design pattern such as building a hierarchy of objects in a simple, descriptive way. This feature helps assembling model applications or model calibration and optimization.

3.1. Components in the Object Modeling System

Components are basic elements in OMS3 which represent self-contained software packages that are separated from the framework. A component can be hierarchical, it may contains other, finer grained components contributing to the larger goal. OMS3 takes advantage of language annotations for component connectivity, data transformation, unit conversion, and automated document generation. A sample OMS3 component to calculate the average of a given vector is illustrated in Listing 1. All annotations start with a `@` symbol.

Listing 1: A sample OMS3 component

```
package oms.components;
import oms3.annotations.*;

@Description("Average of a given vector.")
@Author(name = "Bahman Javadi")
@Keywords("Statistic, Average")
@Status(Status.CERTIFIED)
@Name("average")
@License("General Public License Version 3 (GPLv3)")

public class AverageVector {
    @Description("The input vector.")
    @In
    public List<Double> inVec = null;

    @Description("The average of the given vector.")
    @Out
    public Double outAvg = null;

    @Execute
    public void process() {
          Double sum;
          int c;
          sum = 0.0;
          for (c = 0; c < inVec.size(); c++)
                sum = sum + inVec.get(c);
          outAvg = sum /inVec.size();
    }
```

As one can see, the only dependency on OMS3 packages is for annotations (`import oms3.annotations.*`), which minimizes dependencies on the framework. This enables multi-purposing of components, which is hard to accomplish with the traditional APIs. In other words, components are plain Java objects enriched with descriptive metadata by means of language annotations. Annotations in OMS3 have the following features:

- Dataflow indications are provided by using `@In` and `@Out` annotations.

- The name of the computational method is not important and must be only tagged with `@Execute` annotation.
- No explicit marshaling or un-marshaling of component variables is needed.
- Annotations can be used for specification and documentation of the component (e.g., `@Description`).

For the integration of the OMS3-based workflow environment in the AURIN portal, we have developed a package to generate an html-based documents for each component automatically based on the component annotations [9].

3.2. Model in the Object Modeling System

As mentioned, OMS3 leverages the power of a Domain Specific Language (DSL) to provide a flexible integration layer above the modeling components. To do this, OMS3 gets benefit from the builder design-pattern DSL, which is expressed as a *Simulation DSL* provided by the Groovy programming language. DSL elements are simple to define and use in development of model applications, which is very useful to create complex workflows.

As illustrated in Listing 2, a model/workflow in OMS3 has three parts that need to be specified:

- `components`: to declare the required components;
- `parameter`: to initialize the component parameters;
- `connect`: to connect the existing components.

Listing 2: Model/Workflow template in OMS3

```
// creation of the simulation object
sim = new oms3.SimBuilder(logging:'OFF').sim(name:'test') {
   // the model space
   model {
      // space for the definition of the required components
      components  {
      }

      // initialization of the parameters
      parameter  {
      }
```

```
        // connection of the different components
        connect  {
        }
    }
}
// start of the simulation to obtain the results
results = sim.run();
```

Since OMS3 supports component-based multi-threading, each component is executed in its own separate thread managed by the framework runtime. Each thread communicates to other threads through `@Out` and `@In` fields, which are synchronized using a producer/consumer-like synchronization pattern.

It is worth nothing that any object can be passed between components at runtime. We can also send any Java object as a parameter to the model. Although connecting OMS3 and Java environment is desirable for the AURIN project, we need a method to do it at runtime rather than directly linking Java and OMS3 scripts. Using *GroovyShell* is one way to realize that. In this case, we can bind any object and pass it to the OMS3 scripts. Also, the result of the script can be return as a Java object. More discussions and examples can be found in [9].

4. OMS-based Data-Centric Workflows

In order to create an OMS workflow, basic components need to be provided. The most important components are the Web service clients needed for different service standards; in the case of OGC service, this might be WFS clients, or for statistical data this might be SDMX clients [10]. These clients are used to programmatically access distributed data sets from data (service) providers. To create OMS3 components, there are two main methods to annotate existing codes:

- embedded metadata using annotations;
- attached metadata using annotations;

For the first method, it is necessary to modify the source code (see Listing 1) while for the second one, we can attach a separate file, e.g. a Java class or an XML file for the annotations. Using the attached annotations, we do not need to modify the source code, so the method is well suited for annotation of existing libraries, e.g. common maths libraries can be used as OMS3 components.

In our system, we have developed a package for OMS-based workflows including several OMS3 components, mainly using embedded annotations for the provided components. We also developed several Web service clients with OMS3 annotations to access distributed data provider resources. In the following, we illustrate how we can compose and enact a typical service-oriented and data-centric workflow in the AURIN system.

4.1. Workflow Composition

To create a workflow, it is necessary to either write an OMS script (similar to Listing 2) or save the workflow through the system portal. As users in AURIN are looking for a simple way to compose a workflow, we focus on the second method where users start making some queries through the portal. In this case, they can choose as many datasets as they want and then make the queries through Web service interfaces to get the data as shown in Figure 1. The collected data can be analyzed in the provided portlets in the AURIN portal. At this stage, we can save the current workflow as an OMS3 script. To do this, we developed a package to collect the required parameters for the Web service interfaces used to generate an OMS script. The workflow itself is saved as a text file that can be easily share with other users through the AURIN portal.

An example of an OMS workflow including one WFS client is illustrated in Listing 3. Parameters of this component are automatically generated based on the Web service invocations made through the portal. In this example, the dataset is provided by Landgate WA[9] through its SLIP services[10]. The `bbox` parameter determines the geographical area filter (bounding box) applied to the requested tables (i.e., datasetSelectedAttributes). As seen in this example, DSL makes the workflow very descriptive, which provides flexibility and scalability to generate and share complex workflows.

4.2. Workflow Enactment

To support workflow enactment, we developed a JSR-268 portlet available through the AURIN portal (see Section 2.1). In this portlet, a list of existing workflows is made available that can be executed by users. New workflows can be composed and inserted in this list as well. When a user selects a workflow to run, the execution will be handled by the OMS3 engine.

[9]The provided datasets are from Australian Bureau of Statistics (ABS)
[10]http://landgate.wa.gov.au

Listing 3: An OMS workflow with one WFS client

```
// this is an example for a wfs query
def simulation = new oms3.SimBuilder(logging:'ALL').sim(name:'wfs_test') {

model {

components {
 'wfsclient0'                          'wfsclient'
}
parameter {
 'wfsclient0.datasetName'              'ABS-078'
 'wfsclient0.wfsPrefix'                'slip'
 'wfsclient0.datasetReference'         'Landgate ABS'
 'wfsclient0.datasetKeyName'           'ssc_code'
 'wfsclient0.datasetSelectedAttributes' 'ssc_code,employed_fulltime,
    employed_parttime'
 'wfsclient0.bbox'
     '129.001336896,-38.0626029895,141.002955616,-25.996146487500003'
}
connect {
}
}}
result = simulation.run();
```

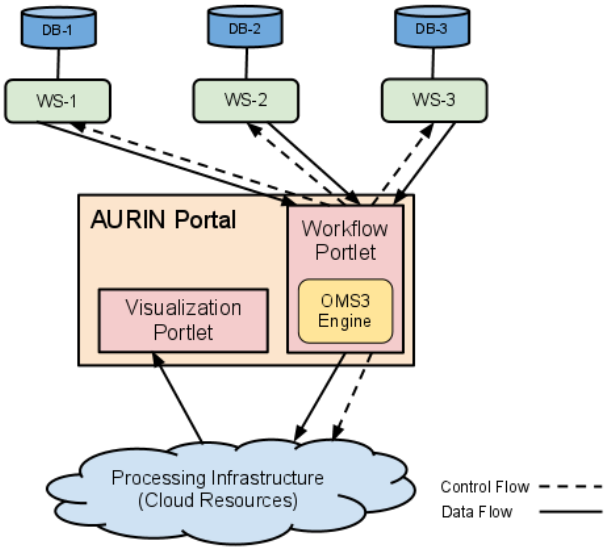


Figure 5: Centralized service orchestration using OMS3 engine.

A sample workflow enactment scenario is illustrated in Figure 5 where WS stands for Web service and DB stands for database. The dashed lines

and solid lines show the control and data flow, respectively. As seen, in this workflow three distributed datasets are accessed through Web services. The workflow portlet then forwards the received data to the processing infrastructure. Finally, the output of processing is sent back to the visualization portlet for user observation. Focusing on the architectural approach of the OMS-based workflows, it can be seen that this model is based on *service orchestration*, which can be a bottleneck to the performance of data-centric workflows.

As we are dealing with data-centric workflows, the output of a service invocation should be ideally directly passed to the processing infrastructure rather than to the centralized engine. To address this, we take advantage of the OMS3 architecture, which is deliberately designed to be flexible and lightweight. To do this, we utilize the OMS3 core and a command-line interface that includes a workflow script and libraries of annotated components to execute a workflow. In many respects, workflow enactment can be thought of as simple execution of a shell script on the command-line. Therefore, when a user requests to enact a workflow from the AURIN portal, the workflow script along with the OMS3 core is sent to the processing infrastructure. In this case, the output of a service invocation can be sent directly to where it is subsequently required in the workflow. This can be considered as a *decentralized service orchestration* or a hybrid model of service orchestration and service choreography. Using this approach, we can decrease the amount of intermediate data and potentially improve the performance of workflows.

Figure 6 shows a decentralized architecture to execute the same workflow as in Figure 5 utilizing a processing infrastructure offered through the Cloud. Here, the data flow is not being passed through the workflow portlet. Rather we delegate the OMS3 core to enact the workflows and receive the data in a place where they are going to be analyzed with computational scalability. Therefore, the decentralized service orchestration can decrease intermediate data and as a result decrease network traffic.

4.3. Cloud-based Execution

Cloud computing environments provide easy access to scalable high-performance computing and storage infrastructures through Web services. One particular type of Cloud service, which is known as Infrastructure-as-a-Service (IaaS), provides raw computing and storage in the form of virtual machines, which can be customized and configured based on application demands [11]. We

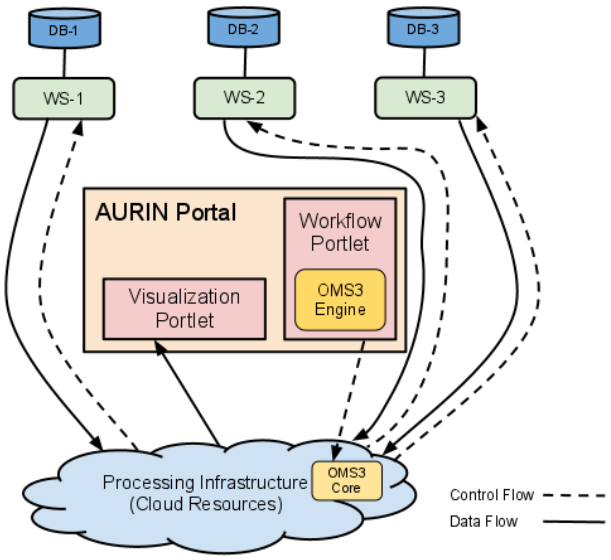


Figure 6: Decentralized service orchestration using OMS3 core.

utilize Cloud resources as the processing infrastructure to execute the complex workflows for both centralized and decentralized approaches.

As noted OMS3 supports parallelism at the component level without any explicit knowledge of parallelization and threading patterns from a developer. In addition to multi-threading, OMS3 can be scaled to run on any Cluster and Cloud computing environment. Using Distributed Shared Objects (DSO) in Terracotta[11], created workflows can share data structures and process them in parallel within a workflow. These features enable us to enact any OMS workflow on Cloud infrastructures as illustrated in Figure 5 and Figure 6.

As discussed in Section 2, the AURIN project is also running in the context of many major e-Infrastructure investment activities that are currently taking place across Australia. One of these projects is the NeCTAR project which has a specific focus on eResearch tools, collaborative research environments, and provisioning of Cloud infrastructures. The NeCTAR Research Cloud [12] offers three types of VMs to Australian researchers:

- *Small*: 1 core, 4GB RAM, 30GB storage
- *Medium*: 2 cores, 8GB RAM, 60GB storage

[11] http://www.terracotta.org/

- *Extra-Large*: 8 cores, 32GB RAM, 240GB storage

At the moment, we use all types of NeCTAR instances as the processing infrastructures based on complexity of the workflows. In addition to the NeCTAR Cloud, we developed an interface to execute the OMS workflows on Amazon's EC2 Cloud offering [13]. This provides an opportunities to utilize Cloud resources from other providers in case of unavailability of the national research Cloud.

5. Performance Evaluation

In order to validate the proposed framework, a set of performance analysis experiments have been conducted. We analyze the execution of some realistic data-centric workflows in the urban research domain on two different Cloud infrastructures.

5.1. Experimental Setup

The workflows that have been considered for the performance evaluation are the initial part of a typical urban analysis task. Spatial data analysis workflows typically start with a data intensive stage where multiple datasets are gathered, and prepared for analysis by building computationally efficient data structures. Most types of spatial analysis include the interrogation of fundamental topological spatial relationships between the constituent spatial objects, such as when two objects *touch* or *overlap* [14]. These relationships fundamentally underpin applications in the spatial sciences, from spatial autocorrelation analysis [15], trip planning [16] and route directions communication [17]. Graph-based data structures are efficient representations supporting the encoding of topological relationships and their computational analysis (e.g., least-cost path algorithms [18]).

In our use case, the collection of suburb and LGA (Local Government Area)[12] boundaries for each of the major Australian states are considered as the input datasets. Each boundary is presented as a *geometry* encoded in the Geography Markup Language [19] (an XML encoding of geographic features). The number of geometries for each state are listed in Table 1. The datasets for each individual state originate from the Australian Bureau of

[12]Each LGA contains a number of suburbs.

Table 1: Number of geometries per state in Australia.

State	**No. of Geometries**	
	Suburbs	LGA
Western Australia (WA)	952	142
South Australia (SA)	946	136
Tasmania (TAS)	402	28
Queensland (QLD)	2112	160
Victoria (VIC)	1833	111
New South Wales (NSW)	3146	178

Table 2: Workflows for the experiments.

Workflow	**Data size (MB)**	
	Geometries	Graph
WA	33.02	2.97
WA, SA	66.44	5.90
WA, SA, TAS	119.75	6.30
WA, SA, TAS, QLD	170.35	21.53
WA, SA, TAS, QLD, VIC	244.97	33.90
WA, SA, TAS, QLD, VIC, NSW	399.04	69.43

Statistics (ABS)[13] and are provided through a OGC WFS service provided by Landgate WA (see Listing 3). The series of WFS *getFeature* queries result in individual feature collections (records) for suburbs/LGAs of each state. The result sets are combined into a single feature collection as part of the workflow, and their topology, based on the spatial relationship (i.e., *touch*) have been computed. The result of the workflow is a topology graph representing adjacencies between suburbs/LGAs with a computational task with a complexity of $O(n^2)$ (unless optimized by a spatial index). This graph then serves as a basic structure for further analysis by urban researchers.

Listing 4: An example of OMS3 workflow for the performance evaluation

```
// This is the Scenario 1 (The first row in Table 2)

def simulation = new oms3.SimBuilder(logging:'ALL').sim(name:'scen1') {
model {
components  {
          'wfsclient0'                    'wfsclient'
          'wfsclient1'                    'wfsclient'
          'graph0'                        'geograph'
          'graph1'                        'geograph'
}
parameter  {
          // WA Suburbs
          'wfsclient0.datasetName'                  'ABS-078'
          'wfsclient0.wfsPrefix'                    'slip'
          'wfsclient0.datasetReference'             'Landgate ABS'
          'wfsclient0.datasetKeyName'               'ssc_code'
          'wfsclient0.datasetSelectedAttributes'
              'ssc_code,the_geom'
          'wfsclient0.bbox'               '112.921113952,
              -35.134846436000004, 129.001930016, -13.689492035'

          // WA LGAs
          'wfsclient1.datasetName'                  'ABS-079'
          'wfsclient1.wfsPrefix'                    'slip'
          'wfsclient1.datasetReference'             'Landgate ABS'
          'wfsclient1.datasetKeyName'               'lga_code'
          'wfsclient1.datasetSelectedAttributes'
              'lga_code,the_geom'
          'wfsclient1.bbox'               '112.921113952,
              -35.134846436000004, 129.001930016, -13.689492035'

}
connect  {
          'wfsclient0.outFC'              'graph0.inFC'
          'wfsclient1.outFC'              'graph1.inFC'
}
}}
```

[13] http://www.abs.gov.au/

```
result = simulation.run();
return [result.graph0.outGraph, result.graph1.outGraph]
```

The series of test workflows based on the aforementioned scenarios is listed in Table 2 where each workflow generates a topology graph for a different number of Australian states. Moreover, the size of input geometries and output graph for these workflows reveal that they are good examples of realistic data-centric workflows. The OMS-based workflow for the first workflow of WA state is illustrated in Listing 4.

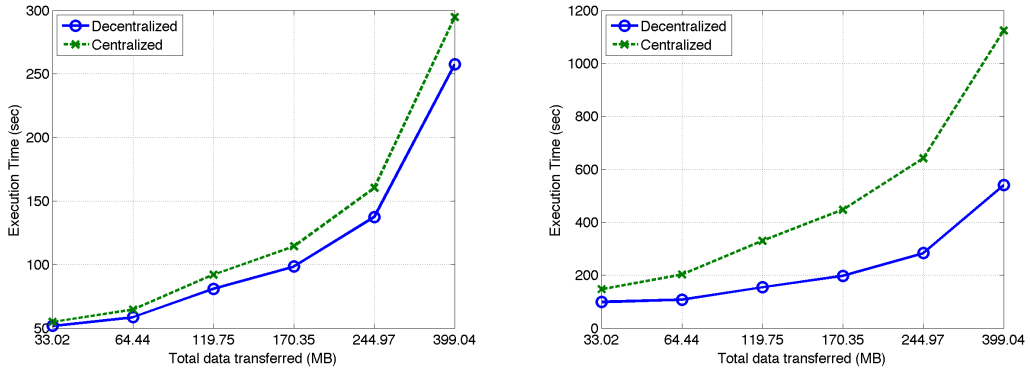
The AURIN portal has been deployed in VMs hosted by NeCTAR NSP, and for each experiment, we enact the workflow on a Cloud infrastructure through this portal. We utilize *Extra-Large* instances from NeCTAR Research Cloud and *Hi-CPU Extra-Large* instances from Amazon's EC2 [13][14]. The characteristics of these two instances in terms of CPU power, memory size, and operating system (i.e., Linux) are similar (see Section 4.3). Each workflow was executed 50 times on both Cloud infrastructures where results are accurate within a confidence level of 95%.

5.2. Results and Discussions

The experimental results for the centralized and decentralized approach for given workflows on the NeCTAR and EC2 Cloud are depicted in Figure 7. In these figures, the y-axis and x-axis display execution time and the total data transferred to the Cloud resources for each workflow listed in Table 2, respectively. It should be noted that in both architectures, the result of the workflow enactment (i.e., topology graph) must be returned to the AURIN portal, so is not included in these figures.

These figures reveal that decentralized service orchestration reduces the workflow execution time in all cases compared to centralized orchestration. For the case of the EC2 Cloud (Figure 7(b)), we can observe a more significant difference between the two architectures, due to limited network bandwidth in Amazon instances. Therefore, decreasing the network traffic using decentralized architecture substantially reduces the execution time of the data-centric workflows. For the results in Figure 7(a), the system portal and Cloud resources are in the same network domain (i.e., NeCTAR network), so higher network traffic can be handled and less improvements are observed.

[14] We choose Asia Pacific region (ap-southeast) to reduce the network latency.



(a) NeCTAR (b) Amazon's EC2

Figure 7: Execution time of data-centric workflows on two Cloud infrastructures for centralized and decentralized orchestration (Each point corresponds to a workflow).

It should be noted that in our experiments, the Web service provider (i.e., Landgate WA) and NeCTAR Cloud infrastructure are in Australia while Amazon's EC2 resources are in Singapore (*ap-southeast* region). So, the larger network latency is another reason for the higher execution time for a workflow on Amazon's EC2 with respect to the NeCTAR Cloud while using the same orchestration architecture.

To compare the effect of the proposed framework in each Cloud infrastructure, Figure 8 plots the average performance improvement for each workflow enactment on the NeCTAR and EC2 Clouds. As expected, the performance improvement for Amazon's EC2 is much higher due to lower network bandwidth. In addition, we execute theses workflows on EC2 instances in the *ap-southeast* region. Using resources from other regions such as *us-east* or *us-west* will increase this improvement. A decentralized architecture thus provides more flexibility in terms of resource selection compared to the centralized service orchestration, which is highly dependent on the network capacity.

As illustrated in Figure 8, the average performance improvement of decentralized orchestration with respect to the centralized one, using NeCTAR Cloud resources is about 20% when we have more than 150MB data to transfer. This improvement can be more than 100% on Amazon's EC2 for such workflows. The reason for a reduced performance improvement for the case

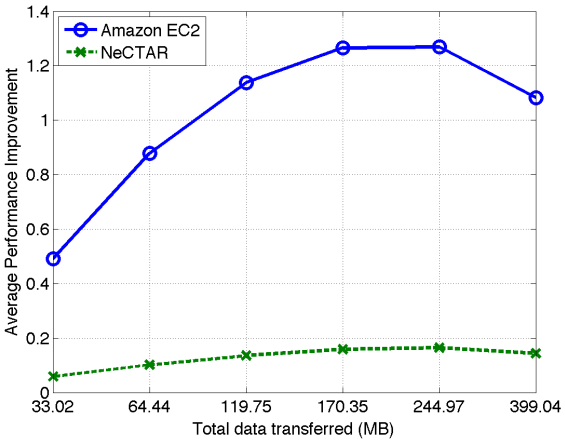


Figure 8: The average performance improvement of decentralized orchestration with respect to centralized orchestration on two Cloud infrastructures (Each point corresponds to a workflow).

of the biggest workflow (i.e., for all states) is the limitation of Web service provider (i.e., Landgate WA) for parallel queries, so the OMS3 engine can not utilize available parallelism in the workflow. This issue disappears if datasets from different data providers are requested in parallel.

Finally, Figure 9 shows the Computation to Communication Ratio (CCR) of the workflows in our experiments. This ratio is commonly used to characterize the distinction between data-intensive and compute-intensive applications [20]. Applications with lower values of this ratio are more data-intensive in nature. As one can see, the value of CCR in our experiments is less than 0.2 for the NeCTAR Cloud and less than 0.1 for EC2. Hence, the workflows are indeed data-centric.

6. Case Study

The AURIN e-Infrastructure currently allows access to a wide range of distributed data sets from a variety of agencies with associated and analytic processing systems. The data providers currently include health related data sets from organisations such as the Public Health Information Development Unit (PHIDU - www.publichealth.gov.au); population statistics data from organisations such as the ABS and delivered by Landgate in Western Australia; econometric data from organisations such as the Centre of Full

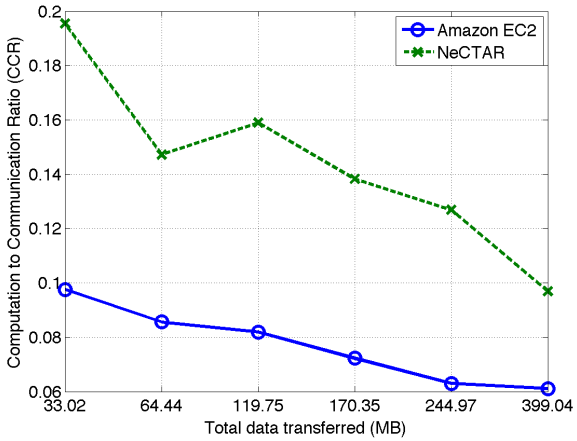


Figure 9: The Computation to Communication Ratio (CCR) on two Cloud infrastructures where each point corresponds to a workflow instance.

Employment and Equity (CofFEE - e1.newcastle.edu.au/coffee); geospatial data from the Public Sector Mapping Agency (PSMA - www.psma.gov.au) - the definitive geospatial information resource for Australia; voting and social demographic data from the Centre for Spatially Integrated Social Science at the University of Queensland through to social media resources such as Twitter feeds.

The typical scenario in accessing data sets in AURIN is to drill into the particular regions of interest, e.g. Melbourne. Through the metadata management systems available in the AURIN infrastructure, filtering of potential data to the site of interest is supported. Researchers typically wish to compare data sets at a variety of aggregation levels, e.g. states, cities, government authorities, statistical local authorities right down to individual cadastre-level (houses). A common phenomenon in AURIN is to support the access to and analysis of data between hitherto separate disciplines. One example of this is the linkage of health data with wider societal data sets. Figure 10 shows the result of searching for data sets associated with accidental deaths in the suburbs of Melbourne and the associated population of those suburbs.

Figure 11 shows the analytical capabilities offered through the AURIN portal including charting and analysis - here showing the correlation between accidental deaths and suicide and the number of accidental deaths for the

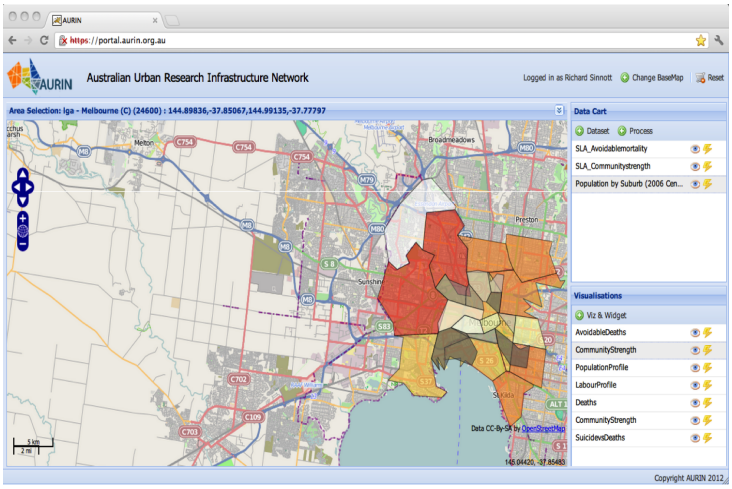


Figure 10: AURIN Portal and Accidental Deaths vs Population Statistics.

various suburbs of Melbourne.

Figure 12 shows a primary driver of the focus of the workflow efforts in AURIN. Specifically establishing a walkability index for particular locations, i.e. how walkable are certain urban locations. This is achieved by firstly selecting points (latitude/longitudes or addresses) of interest (left Figure 12), which when geo-coded using addressing services from PSMA are used to create a network graph based on the street network (centre Figure 12). Using algorithms to calculate the connectivity, the land use mix and the population density, Z-scores are established to give a particular walkability score a given location (right Figure 12). Each of these activities: address/location geo-coding; creating the network buffer; the connectivity/land-use mix/population density establishment, and the final Z-score calculation represent a workflow processing step. In principle, it is possible for the walkability index to be used to create thousands of scores for multiple locations, hence decentralised enactment of processing steps and use of multi-Cloud environments is essential.

7. Related Work

In this section, we present an overview on the related work in orchestration of data-centric workflows. In addition, we compare the OMS framework with the other workflow systems used in the scientific communities.

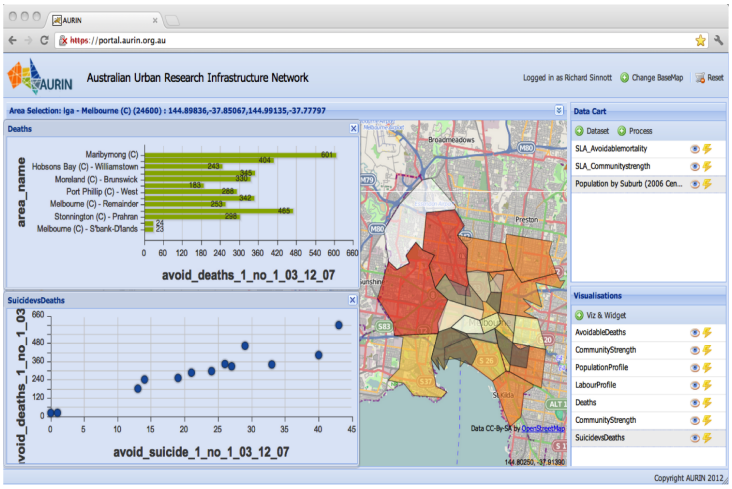


Figure 11: AURIN Portal and Accidental Deaths Analytics.

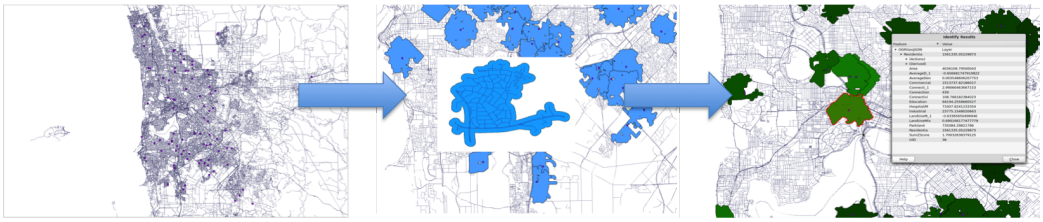


Figure 12: Results of Workflow-oriented Walkability Index.

The most relevant work is from Barker et al. [21, 22], where a *proxy-based* architecture for orchestration of data-centric workflows is proposed. In this architecture, the response to the Web service queries can be redirected by proxies to the place that they are needed for analysis. Although the proposed architecture can reduce data transfer through a centralized engine, it involves deploying proxies in the vicinity of Web services. Moreover, proxy APIs must be invoked by an orchestration engine to take advantage of the deployed proxies. In contrast, our approach does not need any additional component or API calls and can be deployed in any high-performance computing environment as well.

Wieland et al. [23] provide a concept of pointers in service-oriented architectures to pass data by *reference* rather than by value from Web services. This can reduce the data load on the centralized engine and reduce the network traffic. *Service Invocation Trigger* [24] is a decentralized architecture for workflows dealing with large-scale datasets. To utilize this architecture, the input workflow must be first decomposed into sequential fragments without a loop or conditional statement. Moreover, data dependencies must be encoded with the triggers to allow collection of input data before service invocation. In the approach proposed in this paper, a workflow can contain any structure and does not need to be modified prior to execution.

An architecture for decentralized orchestration of composite Web services defined in BPEL is proposed by Chafle et al. [25]. In contrast to our approach, this architecture is very complex and requires code partitioning and synchronization analysis. Moreover, they do not address how these concepts operate in Internet-based Web services.

Other approaches rely on a shared space to exchange information between nodes of a decentralized architecture, more specifically called a tuplespace. In [26], authors transform a centralized BPEL definition into a set of coordinated processes. Through a shared tuplespace working as a communication infrastructure, the control and data dependencies exchange among processes to make the different nodes interact between them. In [27] an alternative approach is presented, based on a chemical analogy. The proposed architecture is composed on nodes communicating through a shared space containing both control and data flows, called the multiset. In contrast, we do not use any shared memory in our proposed framework.

To complete this section, we provide a brief review and comparison of other workflow systems to the OMS framework. Existing workflow systems are designed to utilize the global Grid resources that typically are available for

Table 3: Comparison of workflow languages

Language	**Extensibility**	**Language strength**	**Popularity**	**Libraries**
GSFL	+	-	-	-
GWorkflowDL	+	-	-	+
SCUFL (Taverna)	+	-	-	+
MoML (Kepler)	+	++	++	++
OMS	+	++	+	++

only some specific institutes [28]. Examples are GridFlow [29], Kepler [30], and Pegasus [31]. An overview of these workflow systems is presented in [28]. Although some new projects are started that target the orchestration of workflows using Cloud services, still their applications and overall performance over Cloud infrastructures has not been explored to any great depth. As an example, Cloudbus Toolkit has proposed an architecture for a Cloud-based workflow engine with a centralized orchestration [32]. In contrast, we propose and realize a new framework to implement data-centric workflows with decentralized service orchestration. Some features of the proposed system such as architecture flexibility and multi-threading are unique, which make it a useful Cloud-based workflow engine.

In the following, we also present a comparison of the OMS language with different workflow languages. We considered a set of well-know workflow languages developed for Grid or scientific usage communities. We used the metrics and results of a similar comparison presented in [33]. The results of the comparison is presented in Table 3. *Extensibility* refers to possibility of adding new features to the language. *Language strength* refers to loops and conditions support, data types, and subworkflows. *Popularity* refers to the usage in other projects, available technical support, automatic transformation to other workflow description languages. *Libraries* refers to parsers, graphical representation of workflow.

The list of most common workflow languages are given in the first column of Table 3. As one can see in this table, OMS has many desireable features compared to other languages. The only weakness of the OMS language when compared to MoML and its use in Kepler is in terms of popularity. We hope this paper can encourage researchers to utilize OMS in their projects and thereby rectify this.

8. Conclusions

In this paper, we proposed a new framework to implement data-centric workflows based on the Object Modeling System (OMS). We take advantage of the flexibility of the OMS architecture to implement decentralized service orchestration and thereby bypass the potential bottleneck caused by data flow through a centralized engine. We designed and implemented our proposed framework in the context of the AURIN project to provide a workflow environment for urban researchers across Australia.

Using realistic data-centric workflows from the urban research domain, we evaluated the performance improvement of the proposed architecture whilst utilizing resources from two different Cloud infrastructures: NeCTAR and Amazon's EC2. Performance evaluation results reveal that decentralize service orchestration can substantially improve the performance of data-centric workflows, especially in the presence of network capacity limitations. This work has broad relevance and application to many other disciplines requiring data-intensive research over Cloud infrastructures.

For future work, we intend to extend the evaluation of this architecture using various Web services and network environments to assess the impact of network distance and network configuration. Moreover, we are working on an algorithm to automate provisioning of Cloud resources for data-centric workflows using the OMS framework based on dynamic user demand.

Acknowledgments

We would like to thank the AURIN architecture group for their support. The AURIN project is funded through the Australian Education Investment Fund SuperScience initiative.

References

[1] A. Barker, J. van Hemert, Scientific Workflow: A Survey and Research Directions, in: Seventh International Conference on Parallel Processing and Applied Mathematics, Revised Selected Papers, Vol. 4967 of LNCS, Springer, 2008, pp. 746–753.

[2] T. O. Committee, Web services business process execution language (WS-BPEL), Tech. Rep. Version 2.0 (2007).

[3] N. Kavantzas, et al., Web services choreography description language (WS-CDL), Tech. Rep. Version 1.0 (November 2005).

[4] O. David, J. Ascough II, G. Leavesley, L. Ahuja, Rethinking modeling framework design: Object Modeling System 3.0, in: International Congress on Environmental Modeling and Software, 2010.

[5] B. Javadi, M. Tomko, R. O. Sinnott, Decentralized orchestration of data-centric workflows using the object modeling system, in: CCGRID, 2012, pp. 73–80.

[6] J. Ascough II, O. David, P. Krause, M. Fink, S. Kralisch, H. Kipka, M. Wetzel, Integrated agricultural system modeling using OMS 3: component driven stream flow and nutrient dynamics simulations, in: International Congress on Environmental Modeling and Software, 2010.

[7] R. O. Sinnott, G. Galang, M. Tomko, R. Stimson, Towards an e-infrastructure for urban research across Australia, in: 7th IEEE International Conference on e-Science, 2011, pp. 295 – 302.

[8] A. Panagiotis, Web feature service (WFS) implementation specification, OGC document (2005) 04–094.

[9] B. Javadi, M. Tomko, R. O. Sinnott, AURIN workflows environment using the Object Modeling System, Technical report, Melbourne eResearch Group, The University of Melbourne, Australia (Jan. 2011).

[10] The SDMX technical specification, Tech. Rep. Version 2.1 (2011).
URL `http://sdmx.org/`

[11] J. Varia, Cloud Computing: Principles and Paradigms, Wiley Press, 2011, Ch. 18: Best Practices in Architecting Cloud Applications in the AWS Cloud, pp. 459–490.

[12] T. Fifield, NeCTAR research Cloud node implementation plan, Research Report Draft-2.5, Melbourne eResearch Group, The University of Melbourne (October 2011).

[13] Amazon Inc., Amazon Elastic Compute Cloud (Amazon EC2), `http://aws.amazon.com/ec2`.

[14] M. J. Egenhofer, A formal definition of binary topological relationships, in: W. Litwin, H. Schek (Eds.), 3rd International Conference on Foundations of Data Organization and Algorithms, Vol. 367, Springer-Verlag, 1989, pp. 457–472.

[15] A. Can, Weight matrices and spatial autocorrelation statistics using a topological vector data model, International Journal of Geographical Information Systems 10 (8) (1996) 1009–1017.

[16] M. Duckham, L. Kulik, "simplest paths": Automated route selection for navigation, in: Spatial Information Theory (COSIT 2003), Vol. 2825 of LNCS, Springer-Verlag, 2003, pp. 169–185.

[17] M. Tomko, S. Winter, Pragmatic construction of destination descriptions for urban environments, Spatial Cognition and Computation 9 (1) (2009) 1–29.

[18] P. E. Hart, N. J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Transactions on Systems Science and Cybernetics 4 (1968) 100–107.

[19] C. Portele, Geography markup language (gml3.2.1) encoding standard, specification, Open Geospatial Consortium, Inc. (2007).

[20] S. Pandey, R. Buyya, A survey of scheduling and management techniques for data-intensive application workflows, in: T. Kosar (Ed.), Data Intensive Distributed Computing: Challenges and Solutions for Large-scale Information Management, IGI Global, USA, 2012.

[21] A. Barker, J. B. Weissman, J. van Hemert, Orchestrating Data-Centric Workflows, in: 8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid), IEEE Computer Society, 2008, pp. 210–217.

[22] A. Barker, R. Buyya, Decentralised orchestration of service-oriented scientific workflows, in: CLOSER, 2011, pp. 222–231.

[23] M. Wieland, K. Grlach, D. Schumm, F. Leymann, Towards reference passing in web service and workflow-based applications., in: EDOC'09, 2009, pp. 109–118.

[24] W. Binder, I. Constantinescu, B. Faltings, Decentralized orchestration of composite web services, in: International Conference on Web Services, 2006, pp. 869 –876. doi:10.1109/ICWS.2006.48.

[25] G. B. Chafle, S. Chandra, V. Mann, M. G. Nanda, Decentralized orchestration of composite web services, in: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters, New York, NY, USA, 2004, pp. 134–143.

[26] M. Sonntag, K. Grlach, D. Karastoyanova, F. Leymann, M. Reiter, Process space-based scientific workflow enactment, International Journal of Business Process Integration and Management IJBPIM Special Issue on Scientific Workflows 5 (1) (2010) 32–44.

[27] H. Fernandndez, T. Priol, C. Tedeschi, Decentralized approach for execution of composite web services using the chemical paradigm, in: 2010 IEEE International Conference on Web Services (ICWS), 2010, pp. 139 –146. doi:10.1109/ICWS.2010.46.

[28] J. Yu, R. Buyya, A taxonomy of scientific workflow systems for grid computing, ACM SIGMOD Record 34 (3) (2005) 44–49. doi:10.1145/1084805.1084814.

[29] J. Cao, S. A. Jarvis, S. Saini, G. R. Nudd, Gridflow: Workflow management for grid computing, in: 3rd IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2003), 12-15 May 2003, Tokyo, Japan, IEEE Computer Society, 2003, pp. 198–205.

[30] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, Y. Zhao, Scientific workflow management and the kepler system, Concurrency and Computation: Practice and Experience 18 (10) (2006) 1039–1065.

[31] E. Deelman, G. Singh, M. hui Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, D. S. Katz, Pegasus: a framework for mapping complex scientific workflows onto distributed systems, Scientific Programming Journal 13 (2005) 219–237.

[32] S. Pandey, D. Karunamoorthy, R. Buyya, Workflow engine for Clouds, in: R. Buyya, J. Broberg, A.Goscinski (Eds.), Cloud Computing: Principles and Paradigms, Wiley Press, USA, 2011.

[33] NEXPReS, Workflow description language research results, Technical report, Poznan Supercomputing and Networking Center, Poland (2011).