Decentralized Orchestration of Data-centric Workflows Using the Object Modeling System

Bahman Javadi*, Martin Tomko† and Richard O. Sinnott*
*Melbourne eResearch Group
Department of Computing and Information Systems,
The University of Melbourne, Australia
†Faculty of Architecture, Building and Planning,
The University of Melbourne, Australia
Emails: {bahmanj,tomkom,rsinnott}@unimelb.edu.au

***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 not suitable to execute data-centric workflows. The main issue is transferring output of service invocations through a centralized orchestration engine to the next service in the workflow, which can be a bottleneck for the performance of a data-centric workflow. 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 to avoid passing through the centralized engine. The proposed framework is implemented in 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 research disciplines. Performance evaluation results using spatial data-centric workflows show that we can reduce 20% of the workflows execution time while using Cloud resources in the same network domain.**

***Keywords*-Data-centric Workflows, Object Modeling System, Decentralized Orchestration, Cloud Computing**

I. 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. Although the workflow concept was originally introduced for automation of business processes, there is a huge interest from scientists to utilize these technologies to automate distributed experiments. 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.

Two popular architectural approaches to implement workflows are *service orchestration* and *service choreography* [5]. 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 [10]. 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 XML language [17].

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 [11]. 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.

The OMS framework is currently being used to design and implement a range of science models [3]. However, the capability of this framework for 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 to bypass the limitation of centralized data flow. This feature is crucial for data-centric workflows that deal with large quantities of data and data movement where 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)[1] project, which is an initiative aiming to develop an e-Infrastructure supporting research in the urban and built environment research disciplines [20]. 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.

[1]http://aurin.org.au/

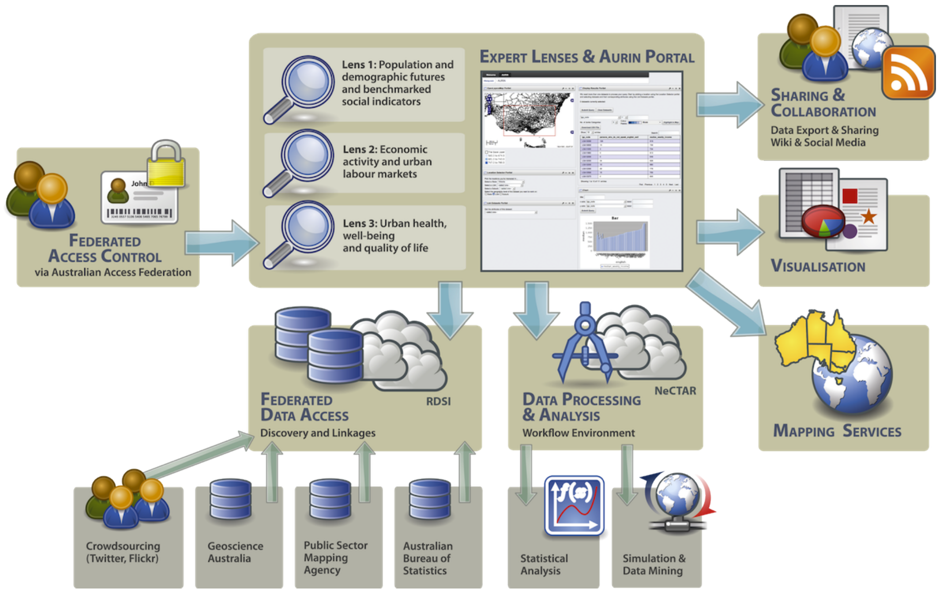

Expert Lenses & Aurin Portal
Lens 1: Population and demographic futures and benchmarked social indicators
Lens 2: Economic activity and urban labour markets
Lens 3: Urban health, well-being and quality of life
Federated Access Control via Australian Access Federation
Sharing & Collaboration
Data Export & Sharing
Wiki & Social Media
Visualisation
Mapping Services
Federated Data Access
Discovery and Linkages
RDSI
Data Processing & Analysis
Workflow Environment
NeCTAR
Crowdsourcing (Twitter, Flickr)
Geoscience Australia
Public Sector Mapping Agency
Australian Bureau of Statistics
Statistical Analysis
Simulation & Data Mining

Fig. 1: The AURIN architecture.

We evaluate the proposed architecture through enactment of realistic data-centric workflows containing data gathering from federated Open Geospatial Consortium (OGC)[2] services and generation of a topology graph for urban analysis. The performance evaluation experiments have been conducted on different Cloud infrastructures to assess the flexibility and scalability of the proposed architecture.

The rest of the paper is organized as follows. We provide an overview of the AURIN project in Section II. In Section III, we present the Object Modeling System framework. Section IV explains the implementation of data-centric workflows using the OMS framework. The performance evaluation of the proposed architecture is presented in Section V. The related work is also illustrated in Section VI. We finally conclude our findings and discuss the future work in Section VII.

II. 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[3] (Figure 1). The sign-on capability is implemented through the integration of the Australian Access Federation (AAF)[4], which provides the backbone for the Internet2 Shibboleth-enabled[5] 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. The 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 a mapping service, and various visualization capabilities.

The federated datasets feeding into AURIN are typically accessed through programmatic APIs. 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 [18] represent one of the most common sources of urban spatial data, served through spatial data infrastructures.

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. These analytical processes allow for advanced statistical analysis of spatial and aspatial data, and also expose complex modeling environments to 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 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.

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

The results of data selection and analysis can be fed to a variety of visual data analytics components, supporting visual exploration of spatio-temporal phenomena. 2D (and soon 3D) 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. Thus will allow researchers from geographically remote locations to collaborate and coordinate on their 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)[6] project, which provides infrastructure services for the research community, and the Research Data Storage Infrastructure (RDSI)[7] 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 NeCTAR Research Cloud as the *processing infrastructure* to execute complex workflows.

III. 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 [11]. OMS version 3.0 (OMS3) provides a general-purpose framework to make easier integration of such models in a transparent and scalable manner. OMS3 is a highly inter-operable 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. Interested readers can refer to [3], [11] for more information about the OMS3 architecture.

The main features of the OMS3 framework are:

- OMS3 adopts a *non-invasive* approach for model or component integration based on annotating 'existing' languages. 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 from and to components. 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. This feature helps assembling model applications or model calibration and optimization.

[6]http://nectar.org.au

[7]http://rdsi.uq.edu.au

A. Components in the Object Modeling System

Components are basic elements in OMS3 which represent self-contained software packages that are separated from the framework. 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 `@` symbol.

Listing 1: A sample OMS3 component

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

@Description("Average of a given vector.")
@Author(name = "Bahman Javadi")
@Keywords("Statictic , 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 (PJO) 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.
- Annotations can be used for specification and documentation of the component (e.g., `@Description`).

In the AURIN application of OMS3, we have developed a package to generate a html-based document for each component, which is itself accessible through the system portal.

B. Model in the Object Modeling System

As mentioned before, 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.

A model/workflow in OMS3 has three parts that need to be specified (see Listing 2):

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

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.

IV. OMS-based Data-Centric Workflows

In order to create an OMS workflow, we need to provide some basic components. The most important components are the Web service clients needed for different service standards; in the case of OGC service, this might be WFS client, or for statistical data this might be SDMX client [1], which are used to get access to various datasets. To create OMS3 components, there are two main methods to annotate the 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 the 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 a few Web service clients with OMS3 annotations to access to the distributed datasets. In the following, we illustrate how we can compose and enact a typical service-oriented and data-centric workflow in the AURIN system.

A. *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 and can be easily share with other users through the AURIN portal.

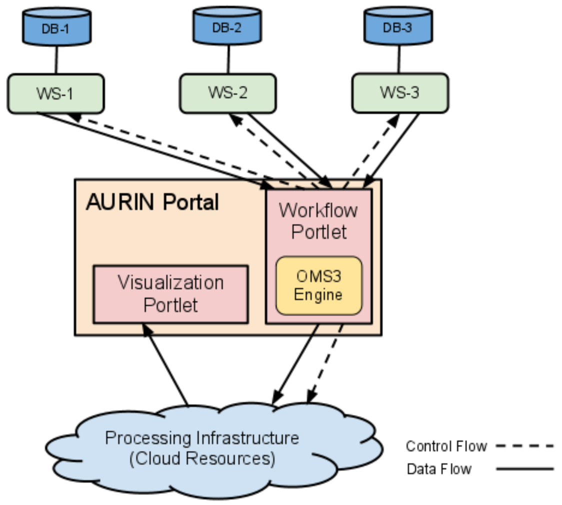


Fig. 2: Centralized service orchestration using the OMS3 engine.

An example of an OMS workflow including one WFS client is illustrated in Listing 2. 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 the Landgate WA[8] through its SLIP services[9]. The `bbox` parameter determines the geographical area filter (bounding box) applied to the requested tables (i.e., datasetSelectedAttributes). As see in this example, DSL makes the workflow very descriptive, which provides flexibility and scalability to generate and share complex workflows.

B. *Workflow Enactment*

To support workflow enactment, we developed a JSR-268 portlet available through the AURIN portal (see Section II). In this portlet, a list of existing workflows is available that can be executed by users. New workflows can also 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.

A sample workflow enactment scenario is illustrated in Figure 2 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 its 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.

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

Listing 2: 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();
```

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 3 shows a decentralized architecture to execute the same workflow as in Figure 2 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 decreases network traffic.

C. 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 services, 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 [23]. We 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

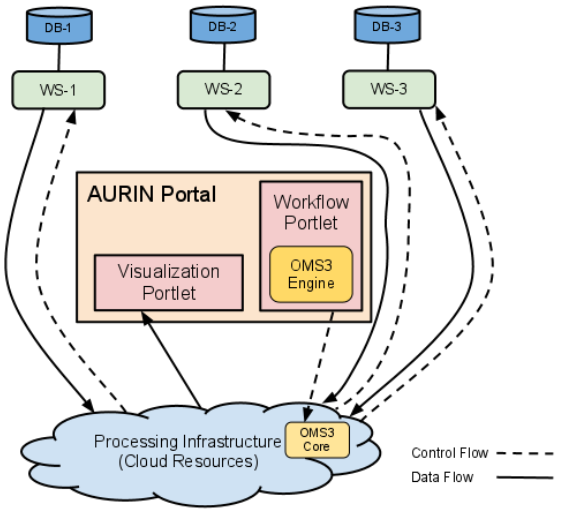


Fig. 3: Decentralized service orchestration using the OMS3 core.

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[10], 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 2 and Figure 3.

As we discussed in Section II, 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 NeCTAR which has a specific focus on eResearch tools, collaborative research environment, and Cloud infrastructure. The NeCTAR Research Cloud [15] is aiming to offer three types of VMs to Australian researchers as follows:

- *Small*: 1 core, 4GB RAM, 30GB storage

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

TABLE I: 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 II: 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

- *Medium*: 2 cores, 8GB RAM, 60GB storage
- *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 NeCTAR Cloud, we developed an interface to execute the OMS workflows on Amazon's EC2 [2]. This provides an opportunities to utilize Cloud resources from other providers in case of unavailability of the national research Cloud. The OMS3 core is very portable and flexible and can be adopted in any Cloud infrastructure.

V. 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.

A. 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* [13]. These relationships fundamentally underpin applications in the spatial sciences, from spatial autocorrelation analysis [8], trip planning [12] and route directions communication [22]. Graph-based data structures are efficient representations supporting the encoding of topological relationships and their computational analysis. (e.g., least-cost path algorithms [16]).

In our use case, the collection of suburb and LGA (Local Government Area)[11] 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] (and XML encoding of geographic features). The number of geometries for each state are listed in Table I. The datasets for each individual state originate from the Australian Bureau of Statistics (ABS)[12] and are provided through a OGC WFS service provided by Landgate WA (see Listing 2). 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.

The series of test workflows based on the aforementioned scenarios is listed in Table II 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 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 [2][13]. The characteristics of these two instances in terms of CPU power, memory size, and operation system (i.e., Linux) are similar (see Section IV-C). Each workflow was executed 50 times on both Cloud infrastructures where results are accurate within a confidence level of 95%.

B. Results and Discussions

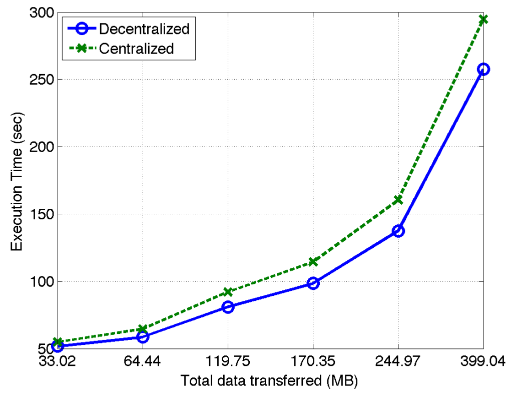
The experimental results for the centralized and decentralized approach for given workflows on the NeCTAR and EC2 Cloud are depicted in Figure 4. In these figures, y-axis and x-axis display execution time and the total data transferred to the Cloud resources for each workflow listed in Table II, respectively. 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 it is not shown 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 4(b)), we can observe 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 4(a), the system portal and Cloud resources

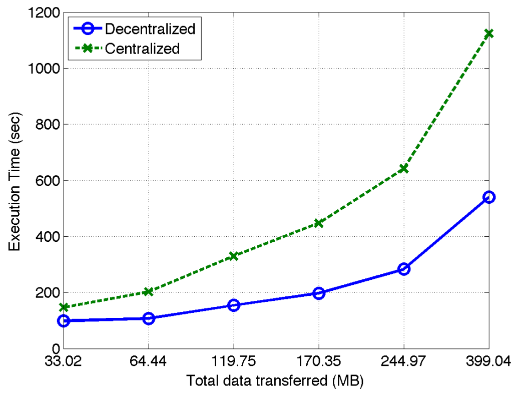
[11]Each LGA contains a number of suburbs.

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

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



(a) NeCTAR (Australia)



(b) Amazon's EC2 (Singapore)

Fig. 4: Execution time of data-centric workflows on two Cloud infrastructures for centralized and decentralized orchestration (Each point corresponds to a workflow).

are in the same network domain (i.e., NeCTAR network), so higher network traffic can be handled and less improvements obtained.

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, larger network latency is another reason of 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 5 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 5, 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 100MB data to transfer. This improvement can be more than 100% on Amazon's EC2 for such workflows. The reason of lesser performance improvement for the case 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 could be disappeared if datasets provided by different Web services are requested in parallel.

VI. Related Work

In this section, we present an overview on the related work in orchestration of data-centric workflows.

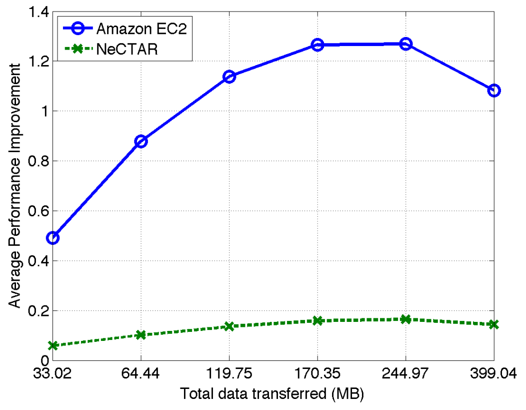


Fig. 5: The average performance improvement of decentralized orchestration with respect to centralized orchestration on two Cloud infrastructures (Each point corresponds to a workflow).

The most relevant work is done by Barker et al. [4], [6], 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. [24] provide a concept of pointers in service-oriented architecture 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* [7] is a decentralized architecture for workflows deal 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. [9]. 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.

Another series of works rely on a shared space to exchange information between nodes of a decentralized architecture, more specifically called a tuplespace. In [21], 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 [14] an alternative approach is presented, based on the chemical analogy. The proposed architecture is composed by 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.

VII. Conclusion

In this paper, we proposed a new framework to implement data-centric workflows based on the Object Modeling System (OMS). Moreover, we take advantage of the flexibility of the OMS architecture to implement the decentralized service orchestration and thereby bypass the potential bottleneck caused by data flow through 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.

For future work, we intend to extend the evaluation of this architecture using various Web services and network environment 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] The SDMX technical specification. Technical Report Version 2.1, 2011.
[2] Amazon Inc. Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/ec2.
[3] J. Ascough II, O. David, P. Krause, M. Fink, S. Kralisch, H. Kipka, and 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.
[4] A. Barker and R. Buyya. Decentralised orchestration of service-oriented scientific workflows. In *CLOSER*, pages 222–231, 2011.
[5] A. Barker and J. van Hemert. Scientific Workflow: A Survey and Research Directions. In *Seventh International Conference on Parallel Processing and Applied Mathematics, Revised Selected Papers*, volume 4967 of *LNCS*, pages 746–753. Springer, 2008.
[6] A. Barker, J. B. Weissman, and J. van Hemert. Orchestrating Data-Centric Workflows. In *8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid)*, pages 210–217. IEEE Computer Society, May 2008.
[7] W. Binder, I. Constantinescu, and B. Faltings. Decentralized orchestration of composite web services. In *International Conference on Web Services*, pages 869 –876, September 2006.
[8] A. Can. Weight matrices and spatial autocorrelation statistics using a topological vector data model. *International Journal of Geographical Information Systems*, 10(8):1009–1017, 1996.
[9] G. B. Chafle, S. Chandra, V. Mann, and M. G. Nanda. Decentralized orchestration of composite web services. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 134–143, New York, NY, USA, 2004.
[10] T. O. Committee. Web services business process execution language (WS-BPEL). Technical Report Version 2.0, 2007.
[11] O. David, J. Ascough II, G. Leavesley, and L. Ahuja. Rethinking modeling framework design: Object Modeling System 3.0. In *International Congress on Environmental Modeling and Software*, 2010.
[12] M. Duckham and L. Kulik. "simplest paths": Automated route selection for navigation. In *Spatial Information Theory (COSIT 2003)*, volume 2825 of *LNCS*, pages 169–185. Springer-Verlag, 2003.
[13] M. J. Egenhofer. A formal definition of binary topological relationships. In W. Litwin and H. Schek, editors, *3rd International Conference on Foundations of Data Organization and Algorithms*, volume 367, pages 457–472. Springer-Verlag, 1989.
[14] H. Fernandndez, T. Priol, and C. Tedeschi. Decentralized approach for execution of composite web services using the chemical paradigm. In *2010 IEEE International Conference on Web Services (ICWS)*, pages 139 –146, July 2010.
[15] T. Fifield. NeCTAR research Cloud node implementation plan. Research Report Draft-2.5, Melbourne eResearch Group, The University of Melbourne, October 2011.
[16] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4:100–107, 1968.
[17] N. Kavantzas and et al. Web services choreography description language (WS-CDL). Technical Report Version 1.0, November 2005.
[18] A. Panagiotis. Web feature service (WFS) implementation specification. *OGC document*, pages 04–094, 2005.
[19] C. Portele. Geography markup language (gml3.2.1) encoding standard. specification, Open Geospatial Consortium, Inc., 2007.
[20] R. O. Sinnott, G. Galang, M. Tomko, and R. Stimson. Towards an e-infrastructure for urban research across Australia. In *7th IEEE International Conference on e-Science*, pages 295 – 302, December 2011.
[21] M. Sonntag, K. Grlach, D. Karastoyanova, F. Leymann, and M. Reiter. Process space-based scientific workflow enactment. *International Journal of Business Process Integration and Management IJBPIM Special Issue on Scientific Workflows*, 5(1):32–44, 2010.
[22] M. Tomko and S. Winter. Pragmatic construction of destination descriptions for urban environments. *Spatial Cognition and Computation*, 9(1):1–29, 2009.
[23] J. Varia. *Cloud Computing: Principles and Paradigms*, chapter 18: Best Practices in Architecting Cloud Applications in the AWS Cloud, pages 459–490. Wiley Press, 2011.
[24] M. Wieland, K. Grlach, D. Schumm, and F. Leymann. Towards reference passing in web service and workflow-based applications. In *EDOC'09*, pages 109–118, 2009.