Elastic Scaling of e-Infrastructures to Support Data-Intensive Research Collaborations

Marcos Nino-Ruiz, Christopher Bayliss, Gerson Galang, Guido Grazioli, Rosana Rabanal, Martin Tomko, Richard O. Sinnott
Department of Computing and Information Systems
University of Melbourne, Vic 3010, Australia
Contact author: marcosnr@unimelb.edu.au

Abstract— **For many research endeavours, e-Infrastructures need to provide predictable, on-demand access to large-scale computational resources with high data availability. These need to scale with the research communities requirements and use. One example of such an e-Infrastructure is the Australian Urban Research Infrastructure Network (AURIN – www.aurin.org.au) project, which supports Australia-wide research in and across the urban and built environment. This paper describes the architecture of the AURIN infrastructure and its support for access to distributed (federated) and highly heterogeneous data sets from a wide range of providers. We present how this architecture solution leverages the intersection of high throughput computing (HTC), infrastructure as a service (IaaS) Cloud services and big data technologies including use of NoSQL resources. The driving concept in this architecture and the focus of this paper is the ability for scaling up or down depending on resource demands at any given time. This is done automatically and on demand avoiding either under- or over-utilization of resources. This resource-optimization-driven infrastructure has been designed to ensure that peak loads can be predicted and successfully coped with, as well as avoid wasting resources during non-peak times. This overall management strategy has resulted in an e-Infrastructure that provides a flexible, evolving research environment that scales with research needs, rather than providing a rigid (static) end product.**

Keywords— e-Infrastructure, resource optimization, Cloud computing, cloud management, cloud bursting, distributed system

I. INTRODUCTION

Recently big data technologies have led to a staggering amount of scientific data being generated [1]. In order to process this ever-growing data, high throughput computing (HTC) aware architectures are essential to support greater problem-solving resources for scientific workflows [2]. This requires frameworks that emphasize reusability and modularity in their components, to enhance integration and support the overall connectivity of data and compute processes. To achieve this, and to facilitate the sharing of heterogeneous data supporting real-time collaborative tasks between geographically distributed science teams, many organisations leverage distributed computer technologies based on web services [3] combined with best practices of HTC together with Cloud computing [4]. Cloud computing-enabled web service frameworks enable potentially thousands of computers to be utilised to share data, applications and computing capacity to achieve a desirable outcome in a manner transparent to the end-user who ideally only interacts with a single entity [5]. The hybrid implementation of HTC, Cloud and the growing demands of domain-specific big data needs are essential to address accessibility and interoperability of e-Science models and big data-driven research endeavours more generally [6-8].

The availability of computational resources through the Cloud allows scaling and support for elastic (on-demand) use of resources [9]. Often this is realised through Infrastructure as a service (IaaS) Cloud services that allow dynamic creation of a variable number of virtual machine (VM) instances depending of the needs of an application at any given point in time. For many disciplines, the access and use of VM is not the core requirement. Instead the ability to dynamically scale software and data applications (Software-as-a-Service) is paramount. In this model a non-trivial challenge needs to be overcome: balancing peak loads and potential for under utilization of resources that consume energy and infrastructure assets. Specifically supporting given quality of service levels and response time is essential [9-11]. To deliver an e-Infrastructure that provides such capabilities in a cost-effective manner it is important to address the issue of over- and under-provisioning of resources [12], and especially tackling the nature of the resource allocation and consumption whereby the number of instances can change greatly between the profiles of the scientific workflows and data sets being serviced [13]. Whilst some support for dynamic scaling of systems exist from major Cloud providers such as Amazon Web Services (AWS) with their Elastic Compute Cloud (EC2) product, the process of scaling of computer and data handling in real-time based on actual user need hasn't been shared among the open research community,– especially across inter-Cloud environments and tackling domain-specific research needs of significance.

The focus of this paper is to describe a solution supporting data-driven, elastic and cost-effective scaling of resource usage based upon profiling policies for resource allocation and consumption according to the (live) data sources being serviced. This work has been undertaken to support the research needs of the Australian Urban Research Infrastructure Network (AURIN [1]) project. The $24m federally funded AURIN project supports nationwide research across Australia in and across the urban and built environment research disciplines. The AURIN e-Infrastructure offers a flexible

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

architecture that delivers a data-driven research environment making available secure, access to distributed (federated) and highly heterogeneous data sets from numerous government, industry and academic organisations across Australia through a single data and tool rich research environment.

The remainder of the paper is organized as follows. In section 2 we summarise related work. In section 3 we present the AURIN e-Infrastructure framework and describe key capabilities that it offers to provide seamless, secure access to large-scale distributed and heterogeneous data sets with integrated data analytical capabilities. Section 4 describes the proposed solution architecture. Section 5 describes how this architecture has been implemented and provides performance results. Finally in section 6 we give concluding remarks and identify future work.

II. Related work

Under- and over- utilization of resources has been researched extensively in Cloud computing, and most of the implementation design underlying this paper is inspired by the work cited in this section.

A body of research has been undertaken on Cloud computing and elastic scaling to realise Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) models [9]. In SaaS – which is closest to the work descirbed here, the need to support a variety of clients/providers, referred as tenants was demonstrated in [14]. Here SaaS providers are aware of the diverse nature of their tenants and multi-tenancy support was controlled at different levels: by the application; by the middleware; by the virtual machine (VM), or by the operating system [15, 16]. A framework to deal with these issues supporting hundreds or even thousands of tenants for SaaS applications was presented in [17]. In this model, variability models were proposed to systematically derive customization and deployment information for individual SaaS using Java Management Extensions (JMX) technology – as employed in this papers' solution. In [18] a broker scheme was used to bridge different profile schemes and deployed in a federated manner across multiple cloud providers. [19]. This approach used a queuing system and the Nimbus toolkit[2] to assign job requests and allocate them to VMs until all requests were serviced, following the same elastic principle of resource allocation. Especially related is the work presented in [10] which established formal measurements for under- and over-provisioning of virtualized resources in Cloud infrastructures, offering the possibility of scaling up or down according to tenants being serviced, although such measurements were targeted more towards SaaS rather than IaaS. The authors in [20] created a framework that allowed to fine-tune resource allocation using constant feedback to capture performance interference interactions and permanent control of Quality of Service (QoS), with provision for additional resources as necessary to achieve the performance that clients required. Another source of inspiration was the framework presented in [21], which supported profiling and behaviour-based management of Cloud components using the CloudSim toolkit[3]. This solution offers generic application support for service provisioning to realise flexible and customized management techniques of the whole Cloud stack. A similar concept of an elastic cluster was described in [4], where the combination of HPC, Grid and Cloud Computing were used to create a hybrid architecture that successfully coped with different workloads and different types of job types and policies across the cloud.

Finally, several commercial tools are available that address similar challenges. The Amazon Auto Scaling tool [22] automatically launches or terminates Amazon EC2 nodes based on customized triggers. This follows similar guidelines as the proposed architecture here. However, it is not directly applicable to this research as it is a proprietary technology that depends on Amazon Cloud services. It is also the case that AURIN is focused upon open source software systems development and delivery.

III. AURIN System Overview

The solution put forward here for elastic compute and data scaling dovetails into a greater and more complex architecture that realises the AURIN project. AURIN aims to develop an e-Infrastructure supporting research in the urban research disciplines [23]. This field is very broad and covers population and demographic futures and benchmarked social indicators; economic activity and urban labour markets; urban health, wellbeing and quality of life; urban housing; urban transport; energy and water supply and consumption, and innovative urban design. Each of these areas represents a significant urban research area in its own right. However a key challenge (and research opportunity) is that all of these areas are themselves inter-related. The AURIN platform is tasked with the integration of tools and data sets that allow such inter-related analyses to take place.

At the heart of the AURIN approach is providing programmatic access to *in situ* data sets in a manner that supports the researchers and their associated research processes whilst ensuring that the data access is consistent with policies from the data providers. To achieve this AURIN supports secure federated data access to many of the definitive urban and built environment data sets from multiple data providers from across Australia. These include the Australian Bureau of Statistics (ABS), Departments of Transport, Departments of Health (amongst many others) as well as a range of data sets from industry stakeholders such as the Public Sector Mapping Agency (PSMA) – the definitive geospatial provider for Australia, and Fairfax Australian Property Monitors (APM) to name a few. The data driven framework realised by AURIN is described in more detail in [23].

In addition to the above, the AURIN project leverages the federally funded ($56m) National eResearch Collaboration Tools and Resources (NeCTAR) Research Cloud (www.nectar.org.au) as its Cloud provider. NeCTAR provides an Australia-wide Cloud infrastructure leveraging the OpenStack Cloud middleware. At present Cloud nodes / resources have been made available in Melbourne (3840 cores/20Tb storage), Monash (2560 cores/10Tb storage), Brisbane (3808 cores/17Tb storage), Canberra (3200

[2] http://www.nimbusproject.org/

[3] http://www.cloudbus.org/cloudsim/

cores/13Tb storage), Adelaide (2992 cores/13Tb storage) and Tasmania (320 cores/2Tb storage). In addition to this, the federally funded ($60m) Research Data Storage Infrastructure (RDSI – www.rdsi.uq.edu.au) is rolling out major storage capabilities across Australia. This is expected to provide over 100 Petabyte data storage for Australian researchers.

The (highl-level) architecture of AURIN is shown in Fig. 1. A detailed description of the AURIN architecture can be found in [23].

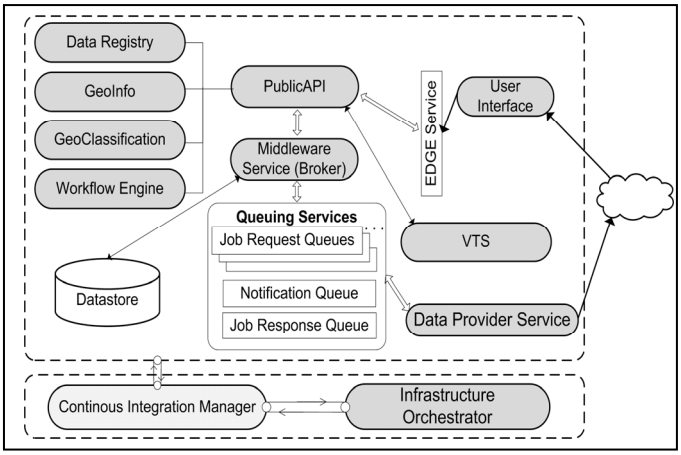

Fig. 1: Overall AURIN Architecture

In summary, the AURIN e-Infrastructure is composed of the following components/modules:

- User Interface: is delivered through a targeted research portal offering seamless and secure access to all of the data sets and tools through a single point of access. The user interface makes heavy use of a range of Javascript libraries including Ext.js, Processing.js, Node.js and Ajax. All user access to the AURIN environment is through a web browser, i.e. there are no client side software requirements.
- VTS: this service is a Map Server that provides vector tiling support for the infrastructure to deliver fast rendering of topographic maps in the UI.
- Edge service: this service provides and supports fine-grained authentication and authorization – leveraging federated access control system information offered through Australian Access Federation (www.aaf.edu.au).
- Public API: provides a front end for the internal modules and components realised the e-Infrastructure – this includes interfaces to the user interface.
- Middleware: is responsible for the control flow of information between the various components, i.e. the business logic.
- Queuing services: provides support for asynchronous communications to help in scalability.
- Data registry: provides detailed information on the data sets and the metadata associated with the range of data providers offering resources to AURIN. At present AURIN makes available over 700 national data sets with rich metadata on the data sets down to the individual attributes associated with any given data set;
- Geoinfo: enables the use of geographical boundaries across multiple resolutions. This supports providing dynamic data interrogation on larger datasets, e.g. disaggregated data across all of Australia.
- Geoclassification: supports a range of geospatial aggregation information across Australia. This includes graphs of spatial information that allow for example to establish all of the local government areas in a given State. There are many geo-classifications that have been put forward by different organizations and these have evolved over time (e.g. Melbourne in 2006 has a different spatial footprint to Melbourne in 2011 and the polygons reflecting the local authority boundaries across Victoria have also changed over time).
- Workflow Engine: provides an analytical toolbox that incorporates an extensive range of analytical tools. These analytical tools utilise Cloud resources. More information on the AURIN workflow environment is given in [28].
- Datastore: provides a NoSQL store for the (heterogeneous) data sets returned from federated data providers. This has been realised using CouchDB[4].
- Data Provider Service: this module is tasked with providing all heterogeneous data requested by the AURIN infrastructure. It is the focus of this paper and explained in detail in the next section.

The ability to optimally use these resources is essential for non-computing science domain scientists - such as the urban and built environment research community.

IV. Data Provider Architecture

A. Overall description

The main contribution of this paper is achieving an optimal balance between deployed capacity and efficient utilization of infrastructure. The driving guideline is to ensure quality of service for a given load of job requests without over-committing memory and wasting resources. It is important to ensure that peak loads can be predicted and successfully coped with, as well as avoid wasting resources during non-peak times. This is essential in the context of AURIN both since it is a national project but also because any given user can start processes that in principle could incapacitate the systems. At the time of writing there have been over 14,000 users of the AURIN research platform from both academia and industry.

The Data Provider module architecture shown in Fig. 2 takes into account four main objectives: accounting for current and expected workloads; ensuring performance; providing stability to ensure QoS constraints are met, and continuous integration of dynamic deployment for new requirements. The latter being of paramount importance in AURIN since the project is an ongoing effort and new requirements, workflows and data sources continually arrive. Each of these requirements result in significant and largely unpredictable scientific

[4] http://couchdb.apache.org/

computing workloads. In this context, the use of virtualization and resource time-sharing may introduce significant performance penalties [24]. Given that HTC and Cloud computing strengths complement each other [4], the former being highly efficient for scientific and diverse analyses that requires strong computing power and resources, and the latter allowing to dynamically grow as/when required, a solution that allocates resources leveraging both strategies is highly desirable.

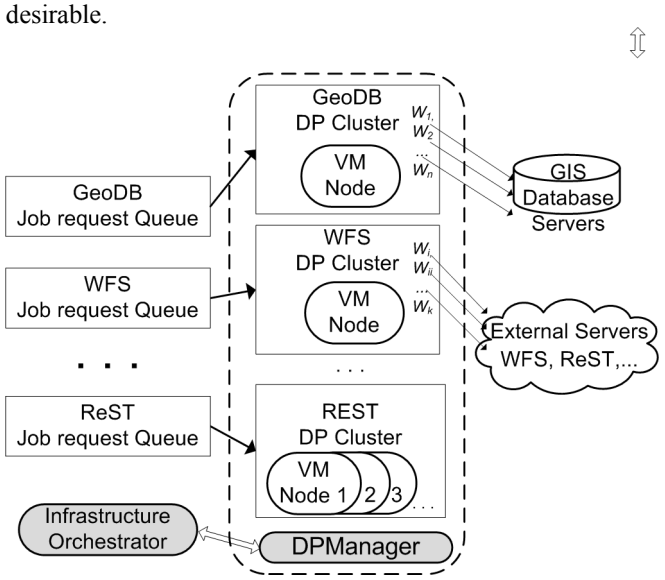

Fig. 2. Data Provider Architecture

The core components of this solution are built using open source technologies. The Data Provider Service is realised through a Java-based platform instrumented through the Spring API[5]. This component implements the JMX technology to provide information on performance, current status and resource allocation to any subscribed listener. It also supports change of runtime configurations on demand. The infrastructure orchestrator uses the Rundeck[6] technology to automatically manage and give non-intrusive support for metering applications, monitoring of deployed applications and providing status information on virtual machine resources. To support the ongoing development of different requirements from AURIN, a continuous integration module (CI) has been realised. This performs quality control of code provided both by internal (in Melbourne) and by external collaborators of AURIN[7]. This includes running automated tests and deployment of the whole infrastructure continuously across four different environments: a nightly build, a development version (used predominantly by the software development teams for rapid prototyping), a staging (pre-production system), and the actual live production that is deployed and used by the urban research community. The continuous integration module (CI) uses the Jenkins[8] technology and supports dynamic reloading of new components that have been tested in the various stages of development. Code peer revision is also realised and supported through the Gerrit review system[9].

The architecture has been designed so that an adequate number of managed resources is available to support workloads (job requests) executing on resources according to a set of scheduling policies based upon previous experience of platform traffic and peak utilisation information. This information is garnered through the portal (production system) that is deployed and active use of this system through targeted workshops (classes of students/researchers) applying and using the system.

B.Elastic Cluster for scaling up / down resources.

The building block of this architecture is the Data Provider cluster (*DPCluster*). The main purpose of a cluster is to execute jobs requests such that they satisfy the timing and throughput requirements of the application. As can be seen in Fig. 2, this module is composed of clusters of workers that are subscribed to job request pipelines or queues (using the ActiveMQ[10] technology). Depending on the type of data source, this cluster can be deployed on one VM node, or as many as are needed according to the processing load and profiling systems in place. When the Data Provider Manager (described later in this section) determines (by the given rules) that another node is required, it performs a ReST service request to the infrastructure orchestrator to create a new one. The size of each *DPCluster* is determined by different factors related to capacity, performance, popularity of the data sets and security requirements. The queuing service ensures that all jobs will be addressed asynchronously as soon as a resource becomes available, without risking inconsistent states for certain job requests, e.g. if the particular type of resources requested are too big or time-consuming. The AURIN middleware service acts as a broker whereby it queues the requests for data sent by the other components of AURIN (UI, external clients, Workflow Engine, among others). The receiver end of these queues is the AURIN Data Provider Service. As can be inferred from previous explanation, the Data Provider Service is composed of a collection of *DPClusters* and a Data Provider Manager. Each cluster is composed of data source-specific workers, which process one job at a time. Specifically they parse data sent through a schema written in JavaScript Object Notation (JSON), which is attached to the payload of the Java Messaging System (JMS) message. Then they build the appropriate query for the external clients; wait for the appropriate response; process returned data and after validation and filter post processing, they ensure that the data conforms to (AURIN) standards for data storage. Finally the Data Provider Service finishes by queuing the resulting data set in the (CouchDB) Data Store queue. All this processed is closely monitored and maintained by the Data Provider Manager component. This component monitors resource allocation as a function of memory at any given time. Every time a cluster is deployed the manager registers it according to a preconfigured profile. The strategy is to fine-tune these profiles (e.g. based on typical use per day, per night, per workshop). Each of them is further divided into specific

[5] http://projects.spring.io/spring-framework/
[6] http://rundeck.org/
[7] There are over 60 separate subprojects in AURIN involving collaborators from across Australia.
[8] http://jenkins-ci.org/
[9] https://code.google.com/p/gerrit/
[10] http://activemq.apache.org/

profiles per data source (see table 1). This approach supports a flexible and customized allocation of resources depending on the type and quality of service provided by the corresponding external client (which can vary in the degree of stability, performance and their availability). The manager dynamically monitors current performance indicators, which trigger certain activities when they reach a threshold (see Fig. 3).

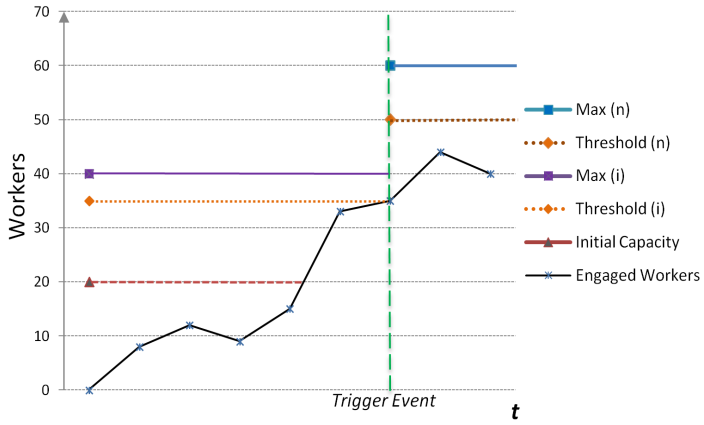

Fig. 3. Strategy for scaling of resources

To define the events determining the thresholds that activate escalation of resources, we use the definition "*point of exhaustion*". When applied within a single node this refers to a limiting resource (e.g. the Java heap space). When this reaches (or nears) 100% utilization for a single virtual machine it identifies that it has reached the capacity that will affect the throughput [12]. The correct identification of these thresholds is essential to measure accurately the potential for resource over-utilization [10]. Conversely, resource under-utilization can be measured by the amount of resources available to be used by potential virtual nodes. In this case it is essential to recognize the moment when a VM node resource usage can be allocated in another node without exceeding the overall limits of resource allocation per node, since that would mean there is resource under-utilization [25]. To recognize when it is appropriate to de-scale the resources committed, we take into consideration the heap memory assigned to the Java virtual machine of the VM node to determine whether it may be selected as a candidate for termination. Each VM is evaluated against the set of deployed VMs on a range of measurements constantly by the DP Manager monitoring service. In the case of memory utilisation, the actual node capacity is determined by subtracting the amount of memory used by the platform from the total Java memory heap size. The removal of a VM node also takes into account the likehood of it being idle for an extended period of time, also factoring in the cost of having a idle node as opposed to powering it down.

To support different profile according to the data source being serviced, each cluster is deployed with an initial and maximum capacity of workers. The workers in this initial pool immediately subscribe to listen to the corresponding job request queue (idle workers). When jobs start to arrive they are assigned to the next available worker, thus becoming active and unavailable until the allocated job is completed (the actual timeframes for this can range from milliseconds to several minutes depending on the use case). When the number of active workers reaches a given threshold, the Data Provider Manager automatically increases the maximum capacity and sets a new threshold to be triggered. This maximum capacity will increase only if enough memory is available; if not, it will send a notification to the Infrastructure Orchestrator. The Infrastructure Orchestrator itself manages the lifecycle of worker nodes, working closely with the Continuous Integration (CI) Module.

V. IMPLEMENTATION

The proposed architecture has been tested using both the web tools available through the Rundeck platform, as well as the *jconsole* metering service, available by default in any Java Development Kit. The snapshots shown in Fig. 4 and 5 reflect the overall behaviour from threads being used on a given node at a given time. The values shown in table 1 also reflect the fact that some of the data sources are more popular than others. It is also the case that the memory footprint for a given job has a close relation with the data itself. Thus whilst benchmarking the computational demands can often be pre-determined, it is the actual utilisation with particular data sets that is often the actual cause of performance bottlenecks.

In Fig. 4 we can see an undesired behaviour of memory usage of a given worker node exhibited during a previous AURIN workshop. Here a given amount of jobs with excessive memory needs are rapidly drawing on the total memory available as indicated by the gradient of the plots in Fig. 4. This results in a fast degradation of effective throughput and negative user experience. This is exhibited by jobs being queued with no available workers to process them, thereby resulting in longer waiting times for the end users. Actual concurrent load and performance of system utilisation in AURIN workshops were used to drive these performance capabilities.

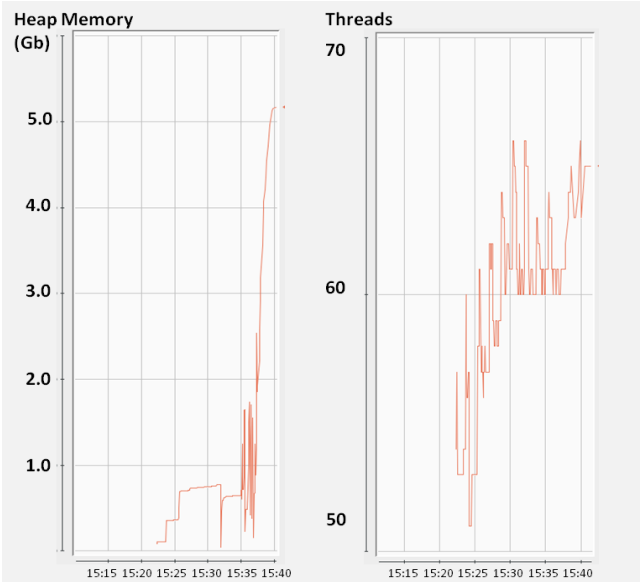

Fig. 4. Java Heap Space being Consumed beyond the Available Capacity

Based on this experience the profiling system is instantiated with initial (expected) configuration information. Thresholds

for given data sources and data processes were established to ensure appropriate response times. These cover: geospatial data access and usage requests (implemented through Open Geospatial Consortium targeted web feature services - WFS); statistical data access requests (Statistical Data Markup Exchange - SDMX) requests; web service requests (utilising Representational State Transfer (ReST services) requests; Graph-based data access requests, and geospatial database access requests (GeoDB). It is important to note that the timing for each of these flavours of data access request is benchmarked on actual data access and use of these flavours of data sets from the research community (obtained during the AURIN workshop). The actual behaviour associated with these data access requests is shown in Table 1.

TABLE 1. PROFILE CONFIGURATION

Data Provider Source type	**High Performance Initial Configuration**				
	Initial pool of workers	***Maximum pool of workers***	***Workers' Threshold***	***Memory Threshold (%)***	***Number of nodes***
REST	3	50	10	90	1
GeoDB	4	5	2	70	5
WFS	5	15	5	80	2
SDMX	5	30	5	90	1
Graph	2	8	2	85	1

Using this configuration information, usage policies and thresholds were defined and applied to accommodate the performance related flexibility (elastic-scaling of the e-Infrastructure). Fig. 5 shows that the Data Provider successfully manages requests to gracefully scale using the Infrastructure Orchestrator to provision new nodes to deal with the incoming job requests before the memory is completely allocated. This ensures that subsequent job requests can be serviced by newly available worker nodes – each of which subscribes to the listening queue associated with the new node.

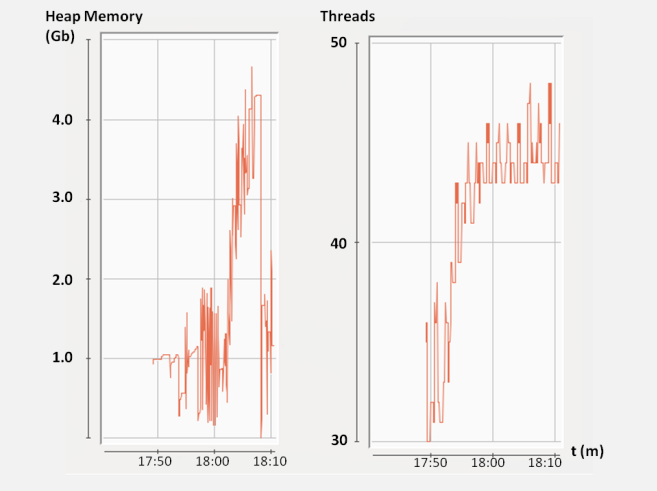

Fig. 5. Dynamic (Elastic-Scaling) to Support Increased Resource Demand

As shown in Fig. 5, the dynamic deployment of new resources based upon increased resource demand illustrates how the system can seamlessly cope with increased resource utilisation. It is important to note that these resources are allocated within a given capacity and overarching threshold that is itself configurable.

VI. CONCLUSIONS

In this paper we have introduced a customized approach to handle resource under- and over utilization when HTC and Cloud platforms are used to support scalable e-Infrastructures to tackle big data challenges. This architecture provides a flexible approach based on complimentary strategies. On the one hand customization of profiles depends on the nature of the data being interrogated and analysed. On the other hand the approach taken supports flexibility of deployment of resources to cope with highly heterogeneous and distributed data sources. Future work revolves around having an automated evaluation module that evaluate performance of *ad hoc* profiles to fine-tune and maximize resource allocation strategies, e.g. implementing resource consuming geospatial analysis that can require extensive and difficult to quantify *a priori* data processing demands. This is particularly relevant considering that more data sets are made available continually in the AURIN e-Infrastructure. Similarly, more workflows and visualization techniques are being implemented continuously in the ever-changing and ever-growing AURIN platform.

In this context, the AURIN project must keep developing and adapting further mechanisms for measuring indices to optimize the determination of trigger points and thresholds for scaling up and down resource usage and associated availability. Currently we have considered a finite set of scalability factors, e.g. memory consumption and a selection of flavours of data sets, however other important measurements such as bandwidth, storage, and latency are increasingly demanded. We have shown that benchmarking performance of such complex systems such as the AURIN platform allows to tackle a wide variety of micro-benchmarks, that can collectively provide a first order estimate of that system's overall expected performance. By the same token, it is important to continue exploring industry standard indices to better inform the monitoring application of the overall performance of this and other modules of the framework [26, 27].

The development and evaluation of the AURIN e-Infrastructure continues fully. A range of new data sets, data flavours (3D, point, polygon, graph-based, etc…) and data providers continues to be made (programmatically) available. Furthermore the computational resources and programmatic APIs for their utilisation continues to evolve. This work thus represents a benchmark for the approach that works for elastic-scaling now. We shall continue to derive new benchmarks and system-level intelligence (profiles) as the project continues.

ACKNOWLEDGMENTS

The AURIN project is funded through the Australian Education Investment Fund SuperScience initiative. The project began in July 2010 and is due to run to end of 2015. We

gratefully acknowledge their support and the wider research community on their feedback.

REFERENCES

[1] A. J. Hey and A. E. Trefethen, "The data deluge: An e-science perspective," 2003.

[2] T. Hey and A. E. Trefethen, "Cyberinfrastructure for e-Science," *Science,* vol. 308, pp. 817-821, 2005.

[3] M. Riedel, "Research advances by using interoperable e-science infrastructures," *Cluster computing,* vol. 12, p. 357, 2009.

[4] G. Mateescu, W. Gentzsch, and C. J. Ribbens, "Hybrid computing—where HPC meets grid and cloud computing," *Future Generation Computer Systems,* vol. 27, pp. 440-453, 2011.

[5] M. Riedel, T. Eickermann, W. Frings, S. Dominiczak, D. Mallmann, T. Dussel, A. Streit, P. Gibbon, F. Wolf, and W. Schiffmann, "Design and evaluation of a collaborative online visualization and steering framework implementation for computational grids," 2007, pp. 169-176.

[6] C. Granell, L. Díaz, and M. Gould, "Service-oriented applications for environmental models: Reusable geospatial services," *Environmental Modelling & Software,* vol. 25, pp. 182-198, 2010.

[7] J. B. Filippi and P. Bisgambiglia, "JDEVS: an implementation of a DEVS based formal framework for environmental modelling," *Environmental Modelling & Software,* vol. 19, pp. 261-274, 2004.

[8] P. Papajorgji, H. W. Beck, and J. L. Braga, "An architecture for developing service-oriented and component-based environmental models," *Ecological Modelling,* vol. 179, pp. 61-76, 2004.

[9] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "A view of cloud computing," *Communications of the ACM,* vol. 53, pp. 50-58, 2010.

[10] J. Espadas, A. Molina, G. Jiménez, M. Molina, R. Ramírez, and D. Concha, "A tenant-based resource allocation model for scaling Software-as-a-Service applications over cloud computing infrastructures," *Future Generation Computer Systems,* vol. 29, pp. 273-286, 2013.

[11] B. Dougherty, J. White, and D. C. Schmidt, "Model-driven auto-scaling of green cloud computing infrastructure," *Future Generation Computer Systems,* vol. 28, pp. 371-378, 2012.

[12] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, "Efficient resource provisioning in compute clouds via vm multiplexing," in *Proceedings of the 7th international conference on Autonomic computing*, 2010, pp. 11-20.

[13] J. L. Lucas-Simarro, R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "Scheduling strategies for optimal service deployment across multiple clouds," *Future Generation Computer Systems,* vol. 29, pp. 1431-1441, 2013.

[14] C. J. Guo, W. Sun, Y. Huang, Z. H. Wang, and B. Gao, "A framework for native multi-tenancy application development and management," in *E-Commerce Technology and the 4th IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services, 2007. CEC/EEE 2007. The 9th IEEE International Conference on*, 2007, pp. 551-558.

[15] C.-P. Bezemer and A. Zaidman, "Multi-tenant SaaS applications: maintenance dream or nightmare?," in *Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE)*, 2010, pp. 88-92.

[16] T. Kwok, T. Nguyen, and L. Lam, "A software as a service with multi-tenancy support for an electronic contract management application," in *Services Computing, 2008. SCC'08. IEEE International Conference on*, 2008, pp. 179-186.

[17] R. Mietzner, A. Metzger, F. Leymann, and K. Pohl, "Variability modeling to support customization and deployment of multi-tenant-aware Software as a Service applications," in *Proceedings of the 2009 ICSE Workshop on Principles of Engineering Service Oriented Systems*, 2009, pp. 18-25.

[18] K. Keahey, M. Tsugawa, A. Matsunaga, and J. A. Fortes, "Sky computing," *Internet Computing, IEEE,* vol. 13, pp. 43-51, 2009.

[19] P. Marshall, K. Keahey, and T. Freeman, "Elastic site: Using clouds to elastically extend site resources," in *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, 2010, pp. 43-52.

[20] R. Nathuji, A. Kansal, and A. Ghaffarkhah, "Q-clouds: managing performance interference effects for qos-aware clouds," in *Proceedings of the 5th European conference on Computer systems*, 2010, pp. 237-250.

[21] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience,* vol. 41, pp. 23-50, 2011.

[22] A. W. Services. (2014, May 2014). *Amazon Web Services—auto scaling*.

[23] R. O. Sinnott, C. Bayliss, G. Galang, P. Greenwood, G. Koetsier, D. Mannix, L. Morandini, M. Nino-Ruiz, C. Pettit, and M. Tomko, "A data-driven urban research environment for Australia," in *E-Science (e-Science), 2012 IEEE 8th International Conference on*, 2012, pp. 1-8.

[24] V. Mauch, M. Kunze, and M. Hillenbrand, "High performance cloud computing," *Future Generation Computer Systems,* vol. 29, pp. 1408-1416, 2013.

[25] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Grid Computing Environments Workshop, 2008. GCE'08*, 2008, pp. 1-10.

[26] P. S. Souza, R. H. Santana, M. J. Santana, E. Zaluska, B. S. Faical, and J. C. Estrella, "Load Index Metrics for an Optimized Management of Web Services: A Systematic Evaluation," *PloS one,* vol. 8, p. e68819, 2013.

[27] A. Iosup, N. Yigitbasi, and D. Epema, "On the performance variability of production cloud services," in *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, 2011, pp. 104-113.

[28] B. Javadi, M. Tomko, R.O. Sinnott, Decentralised Orchestration of Data-centric Workflows in Cloud Environments, *Future Generation Computing Systems,* 2013, http://dx.doi.org/10.1016/j.future.2013.01.008