Capability Analysis of Cloud Resource Orchestration Frameworks

Alireza Khoshkbarforoushha, Australian National University, Australia
Meisong Wang, Australian National University, Australia
Rajiv Ranjan, CSIRO, Australia
Lizhe Wang, Chinese Academy of Sciences, China
Leila Alem, CSIRO Computational Informatics, Australia
Samee U. Khan, North Dakota State University, USA
Boualem Benatallah, University of New South Wales, Australia

Abstract

Since the inception of cloud computing in mid 2000, academic groups and industry vendors have developed a number of Cloud Resource Orchestration Frameworks (CROFs) for simplifying the application management. The CROF aids software engineers, scientists, and infrastructure administrators to migrate and manage their in-house applications to cloud environments. Despite the higher level of technical maturity of such CROFs, there is clear lack of study that can help cloud application administrators in understanding and analyzing the features of CROFs against a common set of concepts and dimensions. Therefore, this study presents a set of generic technical dimensions for clearly analyzing the capabilities of the overriding CROFs. A concise survey and classification of most prominent research work is also presented.

Keywords: *Cloud Computing, Cloud service management, Cloud resource orchestration.*

1. Introduction

Cloud computing assembles a large network of virtualized services, namely: hardware resources (compute, storage, network, etc.) and software resources (databases, message queuing systems, monitoring systems, load-balancers, etc.). The new cloud computing ecosystem enables instant access to virtually unlimited software and hardware resources. It offers a number of considerable advantages including no-upfront investment, lower operating cost, and infinite scalability. However, orchestrating the right set of cloud resources for creating a Quality of Service (QoS) optimized application architecture remains a challenging technical problem. In simple words, the cloud resource orchestration [1] is defined as the process of *Selection*, *Deployment*, *Monitoring*, and *Run-time Management* of software and hardware resources for ensuring that the applications meet the QoS targets, such as availability, throughput, latency, security, cost, and reliability under uncertainties.
To simplify the process of resource orchestration, several academic groups and cloud computing vendors have developed a number of Cloud Resource Orchestration Frameworks (CROFs). For example, Amazon Web Services (AWS) offers AWS Elastic Beanstalk, which is an easy-to-use CROF for deploying and scaling multi-tier Web applications developed with popular programming languages, such as Java, .NET, PHP, Node.js, Python and Ruby. Figure 1 illustrates an instantiation of cloud orchestration operations in AWS.

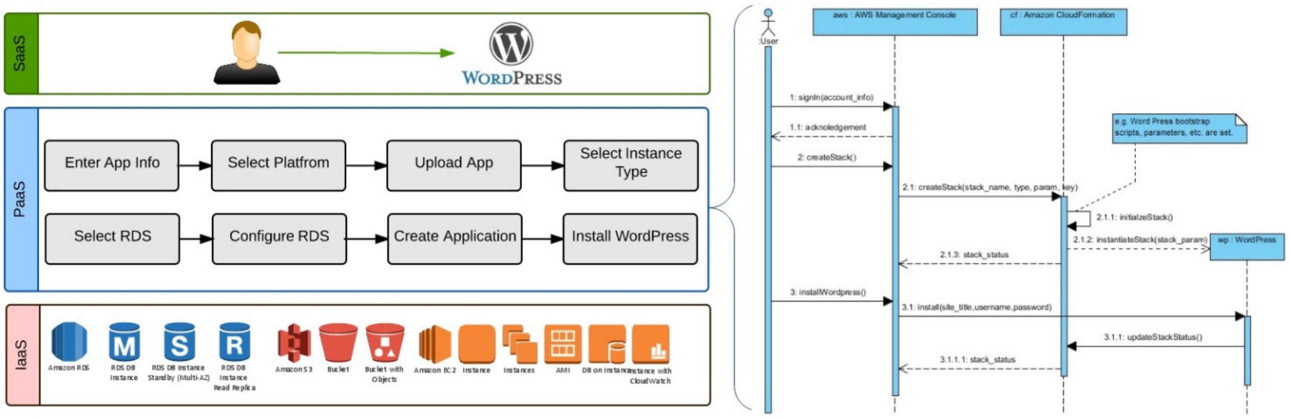

Fig 1: How to deploy WordPress by using either the Amazon BeanSTalk (left hand side workflow) or Amazon CloudFormation (right hand side sequence diagram)

Oayala[1], as another point in case, has developed a CROF for delivering multimedia content online. At the infrastructure layer, Ooyala's CROF leverages AWS EC2 and S3 resources for content distribution and storage, respectively. However, the problem here is that there may exist multiple CROFs from competing vendors and academic groups offering similar (if not the same) set of functionalities. For example, any of the CROFs provided by RightScale, Bitnami, Engine

[1] an international video technology platform and solutions company

Yard, CloudSwitch respectively could be used to manage Web applications over AWS and other public cloud infrastructures. Similar competing CROFs, such as AWS CloudFront, Ooyala, MetaCDN, and Rackspace Cloud Files exist for managing Content Delivery Network (CDN) applications. Therefore, it is clear that the diversity of offerings makes the process (for software engineers, solution architects, or infrastructure administrators) of selecting the most suitable CROF perplexing and a risky task, as wrong decisions could lead to vendor lock-in, poor application QoS, excessive costs, and unwanted administration overheads. Moreover, migration from one cloud to another is nontrivial, if not impossible. Regardless of many cloud standardization projects[2], the community has not yet defined a comprehensive standard that covers all aspects and layers including Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) to mitigate the risk of lock-in for industries [2].

From service providers' perspective, the fact that the existing composition and orchestration techniques have not been adopted for cloud resource orchestration is no accident. The primary goal of most of these techniques has been to achieve better productivity by automating structured processes handled by IT departments or professional programmers, as in large enterprise business processes. Orchestrating cloud resources requires coordination of hardware resources and software resources. In addition to traditional control and data flow programming constructs, there is a need for rich abstractions to support consistency across physical and logical layers, exception handling, and flexible, efficient coordination and management of resources. All of this has to balance the tensions between the rich semantics and simplicity of comprehension, which are the keys to success in practice. Moreover, resource orchestration in cloud environments is complicated due to the scale, heterogeneity, diversity of resource types, and uncertainties of the underlying cloud environment. Apart from heterogeneous software and hardware resources, the integration and interoperation dependencies intensify the challenge of building application architecture over the cloud[1].

The contributions of this study are i) proposing technical dimensions for CROF analysis, and ii) analyzing the strength and weakness of overriding frameworks against the dimensions. We also discuss the recent and most relevant research work concerning each dimension.

2. Dimensions for evaluating Resource Orchestration Frameworks

The diversity of CROFs make the decision making process hard for every software engineers, solution architects, or administrators who wants to migrate their applications to cloud. Having concrete dimensions that give an insight into comparative features of CROFs eases the task of cloud framework selection. We consider the following dimensions Application Domain, Resource Type, Resource Access Component, Interoperability, Resource Selection, Application Deployment, and QoS Adaptation dimensions which evaluate Why, What, and How platforms do resource orchestration tasks:.

- ***Application Domain*****:** This dimension refers to the type of applications that the frameworks have been targeted and customized for including *multi tier web applications* and *CDN*, and *Large-scale data processing (also referred as Big Data).* Multi-tier web application refers to migrating in-house *web*, *mobile*, or *gaming* applications to public or private clouds in order to meet further scalability, availability, etc., conducting *Development and Test* activities on cloud environment, and the like.

  CROFs with CDN application target gives businesses and web application developers an easy and cost effective way to distribute contents (e.g. images, videos, web pages, etc.) to sporadic geographical locations with low latency and high data transfer speeds. To do so, CDN offerings normally use Edge Computing that pushes the frontier of computing applications, data, and services away from centralized nodes to the various servers on the edge of the Internet.

  CROFs with large-scale data-intensive computing platforms are predominantly relying on the simple yet effective programming model, i.e. MapReduce [3]. This model is inherently a divide-and-conquer strategy where a single problem is broken into multiple individual subtasks including various instance of Map and Reduce tasks. This approach is reinforced even more via parallelizing subtasks in a cluster of machines.

- ***Resource Type*****:** This dimension refers to the resource type the framework capable of orchestrating: i) *Infrastructure* or hardware resources such as network (i.e. IP Type, Ports, etc.), CPU (i.e. cores, addressing bit), and BLOB Storage (i.e. storage size, format, etc.); ii) *Platform* resources including application servers (e.g. Java application server), monitoring services, database servers, and the like; iii) *Software* components and sub-processes such as Customer Relationship Management (CRM) business process that is formed and executed via orchestrating software components to deliver a business service to the end users as such the Salesforce CRM[3] offers. These categories are mapped to the so-called cloud service model that is IaaS, PaaS, and SaaS.

- ***Resource Access Component:*** This dimension refers to the mode of interaction with the cloud resources. Interfaces are the high-level abstractions through which an administrator manipulates cloud resources. Currently, there are three types of interfaces supported by resource orchestration frameworks:

[2] http://cloud-standards.org

[3] http://www.salesforce.com/au/sales-cloud/overview/

- ***Low-level command line tool*** that wrap the cloud specific API actions as commands or scripts. These commands can be executed either through Linux (bash or sh) or windows-based (command.com or cmd.exe) shell execution environments. While command line orchestration frameworks can be easier to implement, its usage warrants high-level of technical expertise and understanding about cloud resources and related orchestration operations.
- ***Web based system dashboard*** that represents cloud resources by user-friendly, visual artifacts (buttons, check boxes, etc.), and resource catalogs. Visual artifacts and catalogs aim at simplifying selection, assembly and deployment of hardware and software resources. A catalog manages sets of resource entities that can be instantiated to create CPU, storage, and network objects. System dashboard also features editor where one can assemble and deploy applications by dragging appliance entities from catalog, and integrating and configuring them together via customized configuration management interfaces. Clearly, web based system dashboard offers higher level of simplicity and flexibility to administrators as compared to command line tools.
- ***Web services API*** that enables other tools (e.g. monitoring tools) and systems (e.g. provisioning systems) integrate or use cloud resource management operations into their functionalities. Clearly, Web service API provides the higher level of abstraction and following simplicity. In particular, when some functionalities from the cloud platform has to be integrated to other tools and systems, comparing to the other modes of UI.

- ***Interoperability:*** One of the key barriers for cloud computing adoption is interoperability. Therefore, the platform that has higher level of interoperability with other public or private clouds has a higher chance to be adopted by the industries. To do so, the platforms, more than past, try to expand their multi cloud capability. Consequently, this dimension refers to the ability of a resource orchestration framework to port application across multiple clouds or to use resources from multiple clouds for composing and hosting applications. Interoperability is necessary to avoid cloud provider lock-in. However, designing and implementing generic resource orchestrators that can work with various clouds is nontrivial as it requires APIs specific to each cloud providers.
  We classify the interoperability of orchestration frameworks into *hybrid (or multi cloud)* and *homogeneous (or single cloud)* categories. Hybrid resource orchestrator operates across multiple clouds, transparently integrating their resources (IaaS, PaaS, and SaaS) as part of single resource leasing abstraction. With hybrid orchestrator, administrator can manage and automate the movement of application across clouds as well as communications between resources hosted in different clouds. In contrast, a homogeneous orchestrator is only capable of orchestrating the resources of single cloud provider or set of providers who apply similar technology stack to manage their resources.
- ***Resource Selection:*** This dimension refers to the level of automation supported by orchestration framework as regards to selection of software and hardware resources. Selection process involves identification and analysis of alternatives (cloud resources) based on the preferences of the decision maker (administrator). Making a selection implies that there are alternative choices to be considered, and in such a case administrator not only needs to identify as many of the alternatives as possible but also to choose the one that best fits their selection criteria.
  We classify the resource selection approaches into *Manual* and *Automated* categories. In the manual approach, a resource orchestrator assumes that the administrators have high level of technical knowledge that can help them in selecting the most appropriate clouds resources. In contrast, in an automated selection approach, an orchestrator implements a Recommendation System (RS) [4], [5], [6] that helps the administrator to select the optimal resources that best fits the QoS, feature, and cost needs. The RS ranks different resources based on certain selection criteria and present to the administrator so that they can select the most appropriate ones.
- ***Application Deployment:*** The scale and complexity of applications and cloud resources make them increasingly difficult and expensive to administer and deploy. A recent study of enterprise applications of Fortune 100 companies [7] has revealed interesting facts. The total number of distinct appliances in each application varies from 11 to over 100 depending on the nature of the application. In some of these applications, there are up to 19 distinct front-end web servers, 67 application servers, and 21 back-end databases. Clearly, application deployment technique needs to cater for dependencies across appliances for ensuring error-free configurations. Existing deployment tools provide varying levels of automation, typically categorized as *manual*, *script*-, *language*-, and *model-based* approaches. Higher level of automation in application deployment is preferred for improved correctness, speed, and documentation.
  In the manual approach, appliances are configured and integrated manually by inserting the XML/text snippets into configuration files. Script-based approach consists of a set of shell scripts, which are executed on CPU resources hosting appliances. These scripts directly modify the appliance-specific configuration file. Both manual and script-based approaches have limited ability to express dependencies, react to changes, and verify configurations, which results in erroneous application configuration for large scale deployments. Language-based approaches declaratively define appliance configurations. They provide powerful system modeling capabilities and an expressive notation for describing configurations. Model-based approach defines a desired state for each appliance. Once instantiated on CPU resources, appliances automatically fetch and execute their state definition from centralized repository.

Language and model-based approaches are capable of handling dependencies across appliances and automatically react to changes, such as resource failure, and activating adaptation actions.

- ***Run-time QoS Adaptation:*** This dimension refers to the degree to which a resource orchestrator is able to adapt to dynamic exceptions. Adaptation, in general is realized either *manually* or *automatically*. In a manual manner, provided an event occurs, for example reaching a threshold, the framework does not provide any auto-scaling facility and in the best case it will alert the administrator via an email or message to manually configure the instances in order to accommodate to new conditions. On the contrary, in an automatic fashion the frameworks will adapt to exceptions through some *reactive* and *predictive* techniques. Reactive techniques respond to events only after reaching a predefined threshold that is determined through monitoring the state of hardware/software resources. While these techniques are simple to define and implement (nothing more than an if-then-else statements in nature), they are not sufficient to ensure guaranteed QoS provided a peak demand for resources, for example.
  Predictive techniques can dynamically anticipate and capture the relationship between application's QoS targets, current hardware resource allocation and changes in application workload patterns to adjust hardware allocation. Overall, predictive techniques [6], [8], [9] build upon the integration of theoretical workload prediction and resource performance models. Workload prediction models forecast workload behavior across applications in terms of CPU, storage, I/O, and network bandwidth requirements. The aforementioned models form the basis for the next generation of a dependable resource provisioning framework, in which there is complete understanding of workload and resource demands and therefore improve resilience to uncertainties.

3. Analysis of CROFs

There are a number of CROFs at hand in which we choose 14 important ones that satisfy following conditions; i) covering both proprietary and open source offerings, ii) having reasonable active members iii) having a stable versions, and iv) providing good documentation. This section presents the analysis of these frameworks against the dimensions. Moreover, regarding the laxity of embodiment and acknowledgement of what is proposed in academia by the industry and implementing commercial solutions, we investigate the latest research studies, in particular, in resource selection, application deployment, and run-time adaptation dimensions.

3.1. Application Domain Analysis

Most of the existing off-the-shelf or open source cloud computing platforms do support managing/migrating in-house multi-tire web applications over/to cloud environment. As Table 1 denotes, CloudSwitch[4], CA AppLogic[5], EngineYard[6], Bitnami[7], Amazon BeanSTalk[8], CloudBees[9], RightScale[10] are a number of frameworks in this application category. Our analysis confirms that there are few platforms for other applications as regard to different application types particularly web applications.
RightScale, GoGrid[11], and RackSpace[12] are the frameworks that have large scale data processing target as well; however Amazon and Google offers independent platforms, that is Amazon Elastic MapReduce[13] and Google BigQuery[14]. The solutions for large scale data processing are almost built upon Hadoop[15], an open source framework, in which data and processes are distributed across a resizable cluster of computing server instances. In this regard, once the application and data get ready for processing, it is up to administrator to specify and configure the number and types of computing resources to crunch the deluge of data. This task is very challenging because large-scale data processing platforms (e.g. Hadoop) have numerous configuration parameters (even up to ~200) in which ad-hoc setting significantly impact job performance. Therefore, CROFs should equip users with what-if analysis capabilities through which users will be able to configure and tune respectively cloud resource parameters and data intensive computing platform settings at IaaS and PaaS layers.
Among the analyzed frameworks, GoGrid and Rackspace also offer CDN services in which Rackspace uses Akamai network[16] for delivering contents efficiently to edge nodes. Amazon for CDN application introduces CloudFront[17] that is

[4] http://www.cloudswitch.com/page/enterprise-cloud-computing-product-overview [Accessed on 1/10/2013]

[5] http://www.ca.com/au/cloud-platform.aspx [Accessed on 1/7/2013]

[6] https://www.engineyard.com/products/cloud [Accessed on 1/7/2013]

[7] http://bitnami.com/stacks [Accessed on 13/6/2013]

[8] http://aws.amazon.com/elasticbeanstalk/ [Accessed on 15/3/2013]

[9] http://www.cloudbees.com/ [Accessed on 16/7/2013]

[10] http://www.rightscale.com/ [Accessed on 10/8/2013]

[11] http://www.gogrid.com/ [Accessed on 10/5/2013]

[12] http://www.rackspace.com.au/ [Accessed on 12/6/2013]

[13] http://aws.amazon.com/elasticmapreduce/ [Accessed on 1/11/2013]

[14] https://cloud.google.com/products/big-query [Accessed on 1/11/2013]

[15] http://hadoop.apache.org/ [Accessed on 1/8/2013]

[16] http://www.akamai.com/ [Accessed on 11/3/2013]

integrated with other Amazon Web Services to distribute content to end users with low latency, high data transfer speeds.

3.2. Resource Type Analysis

A majority of cloud frameworks orchestrate virtual appliances (application servers, database servers, etc.) in PaaS layer based on the infrastructure assets of big companies that is Amazon and its offerings such as Amazon EC2[18] and S3[19] for computing and storage requirements, respectively; whereas some others such as Google App Engine (GAE)[20] operates on Google datacenters and Windows Azure[21] that builds upon Microsoft infrastructure services.
Apart from the above-mentioned proprietary IaaS platforms, there exist some open source solutions including OpenStack[22](as serves the IaaS layer of many frameworks such as RackSpace, NeCTAR[23], etc.), OpenNebula[24], CloudStack[25], and Eucalyptus[26]. Amongst these open source IaaS community, OpenStack and CloudStack, both as Apache-licensed cloud computing programs, are most welcome by the user based on their well-defined and documented APIs and also Amazon Web Service. In this regard, a recent study[27] shows OpenStack has the largest active population followed by CloudStack, Eucalyptus, and OpenNebula.

3.3. Resource Access Component Analysis

Resource access component is one of the key capabilities that facilitates the orchestration operations on virtual appliances and to do so, all the under examination frameworks do support web-based interfaces. The provided UIs from cloud frameworks usually do not differentiate between public users and administrators in terms of the simplicity and richness. However, OpenNebula offers various kinds of UI for accessing with virtual computing environment via two different remote cloud interfaces (i.e. web services), OCCI[28] and EC2, and through two web interfaces, OpenNebula Sunstone and OpenNebula SelfService in which the former is for administrators and the latter for public users.
Having Web API is yet another important feature that increases the chance of administrators and developer to employ the orchestrator functionalities into their own tools, if needed. In this regard, as the table 1 shows, most of the under discussion frameworks propose APIs (including REST-full, SOAP-based, etc.), though the functionalities they provide sometimes are more limited than their web UI. For example, EngineYard is working on having the API as a first class citizen on their platform.

3.4. Interoperability Analysis

RightScale is one of the pioneers in meeting interoperability requirement. As table 1 shows, it supports more than ten public or private clouds via its Multi-Cloud Platform through which cloud-specific differences are abstracted so that user can focus on running applications and access the resources via his own terms from either the RightScale Dashboard or API.
CohesiveFT[29] is another most interoperable platform that provides a kind of software factory for assembling and deploying servers to many either public or private cloud platforms. In the same manner, CloudSwitch[30] offer a topology manager that abstracts the details of a cloud provider from the provisioning and management infrastructure required by an application. This means that CloudSwitch could accommodate new providers by allowing their interfaces to be modeled and installed while none of the existing cloud providers get affected.
Amazon Web Services as regard to its dominance and usage span by many users is considered as a target of interoperability of almost every cloud providers. CloudStack, OpenStack and also Windows Azure are supposed to be yet another

[17] http://aws.amazon.com/cloudfront/ [Accessed on 6/11/2013]
[18] http://aws.amazon.com/ec2/ [Accessed on 15/3/2013]
[19] http://aws.amazon.com/s3/ [Accessed on 15/3/2013]
[20] https://developers.google.com/appengine/ [Accessed on 15/5/2013]
[21] http://www.windowsazure.com/en-us/ [Accessed on 10/7/2013]
[22] http://www.openstack.org/ [Accessed on 1/3/2013]
[23] http://nectar.org.au/ [Accessed on 11/10/2013]
[24] http://opennebula.org/ [Accessed on 11/8/2013]
[25] http://cloudstack.apache.org/ [Accessed on 1/11/2013]
[26] http://www.eucalyptus.com/ [Accessed on 4/11/2013]
[27]www.eucalyptus.com/blog/2013/07/03/cy13-q2-community-analysis-openstack-vs-opennebula-vs-eucalyptus-vs-cloudstack [Accessed on 30/10/2013]
[28] Open Cloud Computing Interface: http://opennebula.org/documentation [Accessed on 1/11/2013]
[29] http://www.cohesiveft.com/ [Accessed on 4/11/2013]
[30] http://www.cloudswitch.com/files/CloudSwitch-Enterprise-Data-Sheet.pdf [Accessed on 1/4/2013]

candidate to be targeted for interoperability and as the table 1 denotes they are supported by Bitnami, OpenNebula, EngineYard, and CloudBees.
From another perspective, recent developments including Delta Cloud[31] and JCloud[32] simplify the interoperability task by implementing single API that abstracts APIs related to multiple clouds. For example, Delta Cloud abstracts roughly 15 cloud providers such as Amazon EC2, GoGrid, OpenNebula, OpenStack, Rackspace, Eucalyptus, Terremark[33], and Windows Azure APIs into single API in its recent release. Although aforementioned APIs can simplify implementation across multiple clouds, developers still need to cater for the heterogeneities that prevail in terms of appliance packaging, virtualization technology, resource naming, etc.

Table 1: Mapping of orchestration frameworks to evaluation dimensions

CROF	**Application Domain**	**Resource Type**	**Resource Access Component**	**Interoperability**	**Resource Selection Mode**	**Application Deployment Mode**	**Run-time Adaptation**
CloudSwitch	Web Application (migration)	PaaS	• Web portal • Web service API • Command line	Multi cloud (Amazon EC2, Terremark)	Manual	• Script based • Model based	Manual
RightScale	• Web, Gaming, and Mobile App • Large Scale Data Processing • Development and test	PaaS	• Web portal • API • Command line	Multi cloud (Amazon Web Service, Datapipe, Google Compute Engine, HP Cloud, IDC Frontier, Rackspace, Softlayer, Windows Azure, CloudStack, OpenStack)	Manual	• Script based • Language based	Reactive
CA AppLogic	Web Application	PaaS	• Web portal • Web service API • Command line	Single cloud	Manual	• Script based • Language based • Model based	Reactive
Engine Yard	Web and Mobile Application	PaaS	• Command line • Web portal • API	Multi cloud (Amazon Web Service, Windows Azure, CloudStack, Terremark)	Manual	• Script based • Language based	Manual
Bitnami	Web Application	PaaS	• Command line • Web portal	Multi cloud (Amazon Web Service, Windows Azure, Amazon EC2, RightScale)	Manual	Script based	Reactive
Amazon BeanSTalk	• Web Application • Development and Test • Large Scale Data processing with its Amazon Elastic MapReduce solution • CDN with	PaaS	• Web portal • API • Command line	Single Cloud (Amazon EC2, S3)	Manual	• Manual • Script based (for bootstrapping applications via AWS CloudFormation)	Reactive

[31] http://deltacloud.apache.org/ [Accessed on 4/11/2013]
[32] http://jclouds.incubator.apache.org/ [Accessed on 2/10/2013]
[33] http://www.terremark.com/ [Accessed on 4/11/2013]

	CloudFront solution						
CloudBees	• Web Application • Development and Test	PaaS	• Web portal • API • Command line	Multi Cloud (Amazon Web Service, OpenStack, HP Cloud Services)	Manual	Manual	Reactive
OpenNebula	None (Like OpenStack and Amazon EC2 propose only IaaS services)	IaaS	• Web portal • Web service API • Command line	Multi Cloud (vCloud, OpenStack, Eucalyptus, Amazon EC2)	Manual	• Manual • Model based	Reactive
Eucalyptus	• Development and Test	IaaS and PaaS	• Web portal • REST-based API • Command line	Multi Cloud (Amazon Web Services)	Manual	Manual	Reactive
CohesiveFT	• Web application (migration)	PaaS	• Web Portal	Multi Cloud (IBM SmartCloud Enterprise , Amazon EC2, Amazon VPC, ElasticHosts, Cloud Sigma, Flexiant, Eucalyptus, OpenStack, and vCloud)	Manual	Model-based	Not Known
Google App Engine	• Web and Mobile Application • Development and Test • Large Scale Data processing with BigQuery solution	PaaS (IaaS services are offered using Google Compute Engine)	• Web portal • REST-based API	Single cloud	Manual	Manual	Reactive
Microsoft Azure	• Web and Mobile Application • Development and Test	PaaS	• Web Portal, • Web Services, • Command line	Multi Cloud (Engine Yard)	Manual	Manual	Reactive
GoGrid	• Large Scale Data Processing • Web Application • CDN • Development and Test	IaaS and PaaS	• Web Portal • RESTful API • Command line	Multi Cloud (Windows Azure)	Manual	Manual	Reactive
RackSpace	• Large Scale Data Processing • Web Application • CDN (uses Akamai	IaaS and PaaS	• Web Portal • API • Command line	Multi Cloud (OpenStack)	Manual	Manual	Reactive

	CDN)						

3.5. Resource Selection Analysis

Majority of existing resource orchestration frameworks map to ad-hoc category (Table 1). This means, application composition is prominently up to the administrator who has a great knowledge of balancing loads and demands. Even though, the vast usage of cloud computing technologies forces the industries to work on some smarter approaches such as recommender systems, the existing frameworks, in a short period, could provide some clues for administrators to handle resource selection more precisely. For example, CloudSwitch offers an interesting function, referred to as CloudFit[34], which evaluates the fit of application composer's requested configurations against the available Cloud resource offerings (such as Amazon EC2). CloudFit function ensures that cloud deployments operate with sufficient performance and reliability by automatically selecting the appropriate combination of resources (e.g. processor, memory, and storage). In a broader range, a cloud framework could employ the log files along with the best practices to adjust a fitness function that balance between diverse pool of resources and users requirements as the expert systems do for years in medical issues.

3.5.1. State of the Art in Resource Selection

There are a number of research work that target automatic resource selection. In [4] the authors present a framework that automates the decision-making process based on a model and factors particularly for web server migration to the cloud. Their framework leverages AHP method to automate the selection process based on a model, factors, and QoS parameters. In [5] the authors present a new declarative approach for selecting cloud-based infrastructure services. Their proposed CloudRecommender system automates the mapping of users' specified application requirements to cloud service configurations. In [6] the authors proposed an online resource management decision support system that addresses both tasks scheduling and resource management optimization. The proposed system employs fuzzy and neural network prediction method for predicting VM workload patterns and VM migration time.

3.6. Application Deployment Analysis

In terms of application deployment capability, the CA AppLogic is one step ahead comparing to the others via offering a sound model-based deployment in which an administrator is able to define the workflow of appliances through dragging and dropping the resources one after another. In this manner, user is able to specify the dependency between appliances in an application otherwise the resources will fire all at the same time. Another interesting feature is when you stop an application; appliances will stop in the reverse order of how they started.
CloudSwitch also offers a model-based deployment approach, though in different abstraction level comparing to CA AppLogic. CloudSwitch via Cloud Isolation Technology[35], a sandboxing technology that hides the complexity of public cloud hosting environment from an application, enables administrator to automatically assign Cloud resources to application components, and every piece of data is encrypted end-to-end for achieving high level of security and privacy. Such model based approach guarantees the application will work in the cloud just as if it is in the data center since all the configurations such as IP address, MAC address, and identity will be the same. In the same abstraction level, CohesiveFT enable images to be created and configured dynamically based on preferences which can afterwards be uploaded to cloud infrastructure providers.
Amazon with AWS CloudFormation[36] web service actually complements the deployment process of Amazon BeanSTalk. It automates the details of creating and managing a collection of related AWS infrastructure resources with the aid of i) *Template*, a JSON-format, text-based file that describes all the required AWS resources for running application and ii) *Stack*, the set of AWS resources that are created and managed as a single unit when a template is instantiated by AWS CloudFormation. An interesting feature of CloudFormation is *automatic rollback on error* that guarantees stacks are either fully created, or not at all.

3.6.1. State of the Art in Automatic Deployment

The recent promising study [10] presents a peer-to-peer architecture that uses a component repository to manage the deployment of software components, enabling elasticity by using the underlying cloud infrastructure provider. The provided peer-to-peer architecture has three logical layer; i) Design tier that take the *system description* of the services, ii) Management tier which manages provisioning of services, and iii) Cloud infrastructure tier that enables creation of on demand infrastructure. A proof of concept implementation of the architecture is proposed that in design tier contains a

[34] http://www.cloudswitch.com/page/cloudswitch-architecture-overview-white-paper [Accessed on 5/5/2013]
[35] http://elasticserver.com/ [Accessed on 5/11/2013]
[36] http://aws.amazon.com/cloudformation [Accessed on 6/11/2013]

template designer for both *Eclipse* and *Netbeans* IDEs and also it supports EC2 and an internal HP cloud platform in its infrastructure tier. Since the implementation is open source, it is extensible to new components and features.

3.7. QoS Adaptation Analysis

In terms of adaptation dimension analysis, excluding CloudSwitch and EnginYard cloud frameworks that leave adaptation and scalability of cloud to the administrator or developer through programmatic manual approach, all of the under discussion frameworks are reactive (Table 1). Among these reactive platforms, there are cases in which the platform leaves implementation of auto-scaling and run-time adaptation to the developer via providing different APIs. For example, in GoGrid platform infrastructure scaling process is not automated. That is cloud users are allowed to manually vary (scale or de-scale) configurations of the compute server and storage resources through the customer portal. In case of virtualized compute server, the changes can be only affected for physical memory allocation, while the CPU and local storage allocation remains the same. GoGrid gives developers an option of implementing custom auto-scaling service through its API[37].

3.7.1. State of the Art in QoS Adaptation

In terms of predictive QoS adaptation, the authors in [8] propose prediction-based resource measurement and provisioning strategies using Neural Network and Linear Regression. The prediction method uses historical data that is generated by running a standard client–server benchmark on Amazon EC2 for training forecasting models. Unlike [8] which discusses the generic prediction framework, the authors in [9] employ a predictive models for resource provisioning for read intensive multi-tier applications in the cloud. Yet another work that uses time series methods is [11] in which the authors develop a resource prediction and provision scheme that uses Autoregressive Integrated Moving Average (ARIMA) time series analysis for prediction model.
Another interesting work is [12] in which they present an online temporal data mining system to model and predict the cloud VM demands. In fact, their system extract high level characteristics from VM request stream and notify the provisioning system to prepare VMs in advance.

4. Conclusions

Diversity of cloud management framework offerings makes the process of decision making for software engineers, solution architects, or infrastructure administrators challenging. To address such an issue, this study explores the capabilities of the overriding cloud platforms in terms of following dimensions: application domain, resource type, resource access component, inter-operability, resource selection, application deployment, and QoS adaptation, that altogether shed light on *why*, *what*, and *how* these frameworks do the resource orchestrations operations.

Acknowledgement

Rajiv Ranjan's research at CSIRO is funded by Australia India Strategic Research Grant titled "Innovative Solutions for Big Data and Disaster Management Applications on Clouds (AISRF-08140)". The funding body is Department of Industry, Australia.

References

[1] Ranjan R, Benatallah B: “Programming Cloud Resource Orchestration Framework: Operations and Research Challenges”.19. Technical report, arXiv:1204.2204, Published Online on 10 April 2012

[2] Grace A. Lewis, “Role of Standards in Cloud-Computing Interoperability” IEEE, 46th Hawaii International Conference on System Scieneces, pp. 1652-1661, 2012

[3] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." Communications of the ACM 51.1 (2008): 107-113

[4] Menzel, M.; Ranjan, R.; Wang, L.; Khan, S.; Chen, J., "CloudGenius: A Hybrid Decision Support Method for Automating the Migration of Web Application Clusters to Public Clouds," Computers, IEEE Transactions on , vol.PP, no.99, pp.1,1 doi: 10.1109/TC.2014.2317188

[5] Miranda Zhang, Rajiv Ranjan, Surya Nepal, Michael Menzel, and Armin Haller. 2012. A declarative recommender system for cloud infrastructure services selection. In Proceedings of the 9th international conference on Economics of Grids, Clouds, Systems, and Services (GECON'12), Kurt Vanmechelen, Jörn Altmann, and Omer F. Rana (Eds.). Springer-Verlag, Berlin, Heidelberg, 102-113. DOI=10.1007/978-3-642-35194-5_8

[6] Ramezani, Fahimeh, Jie Lu, and Farookh Hussain. "An online fuzzy Decision Support System for Resource Management in cloud environments." IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS), 2013 Joint. IEEE, 2013.

[7] Hajjat, Mohammad, et al. "Cloudward bound: planning for beneficial migration of enterprise applications to the cloud." ACM SIGCOMM Computer Communication Review 40.4 (2010): 243-254.

[8] Islam, Sadeka, et al. "Empirical prediction models for adaptive resource provisioning in the cloud." Future Generation Computer Systems 28.1 (2012): 155-162.

[9] Iqbal, Waheed, et al. "Adaptive resource provisioning for read intensive multi-tier applications in the cloud." Future Generation Computer Systems 27.6 (2011): 871-879.

[37] http://blog.gogrid.com/2013/04/23/how-to-create-an-auto-scaling-web-application-on-gogrid-part-1-theory/ [Accessed on 1/4/2013]

[10] Kirschnick, Johannes, et al. "Towards an architecture for deploying elastic services in the cloud." Software: Practice and Experience 42.4 (2012): 395-408.
[11] Fang, Wei, et al. "RPPS: A Novel Resource Prediction and Provisioning Scheme in Cloud Data Center." Services Computing (SCC), 2012 IEEE Ninth International Conference on. IEEE, 2012.
[12] Jiang, Yexi, et al. "Asap: A self-adaptive prediction system for instant cloud resource demand provisioning." Data Mining (ICDM), 2011 IEEE 11th International Conference on. IEEE, 2011.

Supplemental Material

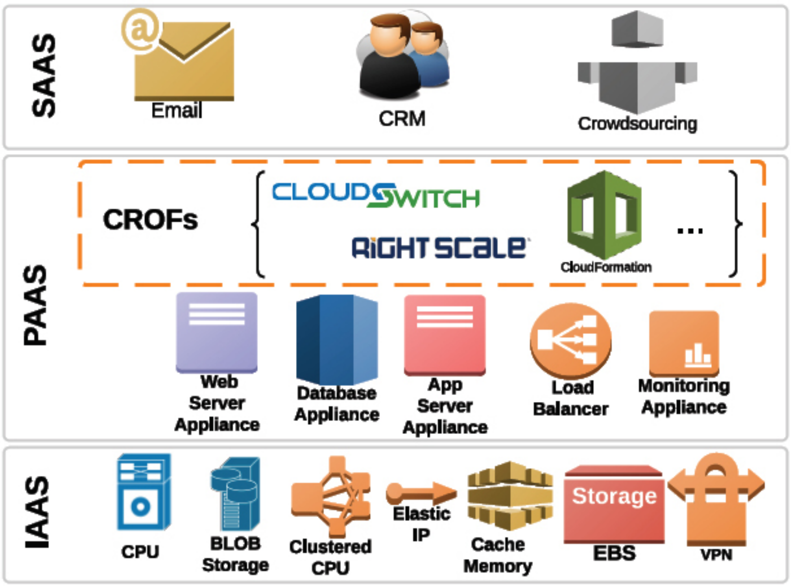

Figure 2: Cloud resource orchestration framework (CROF) position in cloud ecosystem.

Alireza Khoshkbarforoushha is a Ph.D. candidate in the College of Engineering and Computer Science, The Australian National University (ANU). He is also a Graduate Researcher with the Commonwealth Scientific and Industrial Research Organization (CSIRO), Canberra, Australia. His research interests include large-scale data-intensive systems, database-as-a-service, query resource management on cloud, and streaming workload management on cloud.
Contact details: Computer and Information Technology Building (Building 108), Australian National University, North Road Acton ACT 2601, Australia; Tel: 61 2 62167115

Meisong Wang is a Master of Research student in the College of Engineering and Computer Science of ANU and an assistant researcher at CSIRO. His research is around Cloud Computing, distributed TDT (Topic Detection Tracking) based on Apache Hadoop ecosystem such as Mahout, Yarn, Hbase.
Contact details: Computer and Information Technology Building (Building 108), Australian National University, North Road Acton ACT 2601, Australia

Dr. Rajiv Ranjan is a Julius Fellow (2013-2016), Senior Research Scientist and Project Leader in the Digital Productivity and Services Flagship of Commonwealth Scientific and Industrial Research Organization (CSIRO). Rajiv has +100 scientific publications. His h-index is 24, with a total citation count of 3300+.
Contact details: Computer and Information Technology Building (Building 108), Australian National University, North Road Acton ACT 2601, Australia; Tel: 61 2 6216 7047, Fax: 61 2 6216 7111

Dr. Wang is a Professor at Institute of Remote Sensing & Digital Earth, Chinese Academy of Sciences (CAS) and a ChuTian Chair Professor at School of Computer Science, China Univ. of Geosciences (CUG). Prof. Wang received his B.E. & M.E from Tsinghua Univ. and Doctor of Eng. from Univ. Karlsruhe, Germany. Prof. Wang is a Fellow of IET, Fellow of British Computer Society. Prof. Wang leads high geo-performance computing group at CAS and the High Performance Computing Lab at CUG. His main research interests include high performance computing, e-Science, and spatial data processing.
Contact details: No.9 Dengzhuang South Road, Haidian District, Beijing 100094, P.R. China, Phone/Fax:86-10-82178070

Dr. Leila Alem is a principal research scientist at the CSIRO's ICT Centre Information Engineering research Laboratory based in Sydney. Her formal training is in artificial intelligence and cognitive psychology. She has designed and evaluated numerous advanced decision support systems and advanced computer supported training. Over the last 18 years of her career at CSIRO she has designed and evaluated various advanced user interfaces for domains including mining,

aviation, automotive and health. Her current research interest include: Human Computer Interaction, Computer mediated interaction, Evaluation of information systems.
Contact details: Crn Vimiera and Pembroke Roads, Marsfield NSW 2122, Tel: +61 2 9372 4366, Fax: +61 3 9545 8080

Samee U. Khan received a BS degree in 1999 from Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Topi, Pakistan, and a PhD in 2007 from the University of Texas, Arlington, TX, USA. Currently, he is Associate Professor of Electrical and Computer Engineering at the North Dakota State University, Fargo, ND, USA. Prof. Khan's research interests include optimization, robustness, and security of: cloud, grid, cluster and big data computing, social networks, wired and wireless networks, power systems, smart grids, and optical networks. His work has appeared in over 250 publications.
Contact details: Samee U. Khan, Department of Electrical and Computer Engineering, North Dakota state University Fargo, ND 58108, USA

Benatallah is a professor of Computer Science and Engineering at UNSW Australia. His research interests include API engineering, Web services composition, federated cloud services orchestration, crowd sourcing services and business process management. He published over 200 peer-reviewed papers on these topics. He has a PhD from Grenoble University, France. He is member of IEEE and ACM.
Contact details: Prof. Boualem Benatallah, School of Computer Science and Engineering, UNSW Australia, UNSW SYDNEY NSW 2052, AUSTRALIA, Fax: (61) 2 93854767, Phone: (61) 2 93854767