CloudMan as a Tool Execution Framework for the Cloud

Enis Afgan, Davor Davidović, Tomislav Lipić, Ivan Sović and Karolj Skala
Ruđer Bošković Institute/Centre for Informatics and Computing,
Bijenička 54, HR-10000, Zagreb, Croatia
{Enis.Afgan, Davor.Davidovic, Tomislav.Lipic, Ivan.Sovic, skala}@irb.hr

Abstract - Cloud computing has revolutionized how availability and access to computing and storage resources is realized; it has made it possible to provision a large computational infrastructure in a matter of minutes, all through a web browser. What it has not yet solved is accessibility of a tool execution environment where tools and data can easily be added and used in non-trivial scenarios. In this paper, we demonstrate how CloudMan (http://usecloudman.org) can be used to provide complete and complex tool execution environments for making cloud resources functional for a desired domain.

I. INTRODUCTION

Conducting controlled, consistent, verifiable and repeatable experiments is of utmost importance for the development of computer science [1] - although this paradigm is applicable for every research regardless of the field of study. Additionally, many branches of modern research are computation- and data-intensive forms of discovery, encompassing the generation, analysis and interpretation of vast amounts of data against catalogs of existing knowledge in complex multi-stage workflows that are enabled by a combination of analysis platforms and computational infrastructures. Massive amount of computing power, derived from a large number of computers, is often required to perform these large-scale experiments [2]. Scientific computing in most cases requires availability of such distributed computing and networking platforms which, in traditional high-performance computing solutions, account either for locally installed clusters and supercomputers, or distributed grids, that are often difficult to setup, maintain and operate. On the other hand, cloud computing provides a completely new model of utilizing the computing infrastructure, where computing and storage resources can be provisioned in a dynamic, scalable, on demand and pay per use basis, and released when no longer needed [2].

Cloud computing refers both to applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services [3]. In this manner, the cloud can be divided into three major service types that are available to end users, scientific institutions, and enterprises: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [2]. IaaS refers to provisioning IT infrastructure resources over the network based on virtual or physical resources that meet the CPU, power, storage, and operating system requirements [2]. Amazon Web Services (aws.amazon.com) is an example of a major IaaS vendor, while other providers include GoGrid and Flexiscale **Error! Reference source not found.**. PaaS encompasses the provisioning of a platform layer with resources such as software development frameworks for applications that will run on the cloud. Applications utilizing PaaS layer interact with the cloud on the higher level of abstraction while the low-level components are hidden behind the platform layer. Examples of PaaS providers include Google App Engine and Microsoft Azure. SaaS represents the top end of the cloud computing stack by providing functional applications and services but also yielding least user customization [2]**Error! Reference source not found.**. Examples of SaaS include Google Documents, online email clients, or photo sharing sites.

Although gradually adopted by select groups, the potential offered by cloud platforms is only partially exploited because much of its functionality is still cumbersome to use for the end users. The reality of using cloud resources is that the necessary tools, libraries, and applications for every day best practices in research and operational business routines are by themselves relatively complicated to install and customize. Even if those are made available via predefined platforms or custom solutions, their scope is often limited and usage predefined. They typically also involve a high level of on-going maintenance to keep the software and data up-to-date. Such maintenance currently requires significant expertise in software development and system administration. These requirements stand in the way of cloud adoption, particularly among small development and research groups with insufficient capacity to tackle such IT challenges.

Currently many powerful solutions exist for materializing the cloud concepts, but somewhat lack easy-to-use principles that would allow end users to take full advantage of them. Thus, developing cloud methodologies is no longer the main challenge in the uptake of the cloud. Instead the problem lies in making existing methodologies usable for end users so that they can take full advantage of the existing resources. Some of the challenges standing in the way of the widespread acceptance of clouds are being addressed by a number of research projects, contributing to minimize the technical complexities of using the cloud by leveraging and developing cloud platforms to reach higher levels of innovation and automation [5]. These research projects include (amongst others): mOSAIC [6], 4CaaST [7], Contrail [8], CumuloNimbo [9], StratusLab

[10], StarCluster [11], RedHat OpenShift Flex [12], Manjrasoft Aneka [13], and Cirrocumulus [14].

To bridge this critically important gap between computational resources and end users we have previously developed CloudMan [1][15][17]. CloudMan makes it remarkably easy to exploit the potential of cloud resources without burdening users with tedious details: it delivers a framework and a platform for providing accessible solutions built on the flexibility of IaaS while delivering it in a PaaS format. The software and services offered by CloudMan have already attracted and established a growing community of users and developers, which confirms the emerging trend of acceptance and integration of accessible solutions based on the cloud computing model into the everyday workflow of end users.

In this paper we demonstrate the application of CloudMan as a tool execution framework for the cloud, and give insight to the ease and rapidity of deploying configured and ready to use systems. In Section II we present the overview of the CloudMan architecture including supported file systems, the data persistence model, and deployment concepts. Lastly, we describe three categories of applications that can be deployed into the CloudMan platform, thus alluding at the potential flexibility of the described platform.

II. Tool Deployment and Execution Framework for the Cloud

The cloud infrastructure and the overlaying layer for operating it are in a mature phase (e.g. AWS, OpenStack, Eucalyptus, OpenNebula). Although the infrastructure provides most of the required functionality to gain access to a functional infrastructure, it is quite complicated and time consuming for the inexperienced and non-technically educated users to properly exploit infrastructure's full potential. CloudMan has successfully resolved many issues regarding accessibility to cloud resources.

CloudMan provides an interface between IaaS and SaaS, acting as a platform that encapsulates several important functionalities. Some of the most prominent features that are supported include automated (and thus reproducible) configuration for machine image, dynamic persistent storage for tools and data, automated scaling of the underlying infrastructure, and sharing of complete deployments. All this is exposed through a web browser making it very accessible to end users.

The underlying architecture allows each CloudMan deployment to be customized by individual users and thus meet their specific needs. Once customized, the customizations can be persisted even after the cloud instance is terminated (i.e. cloud image can be saved and re-opened again without losing any information). Furthermore, the customized instances can easily be shared with different users within the same cloud provider. This implies sharing of the deployment (including the applications and the data) so that the entire deployment runs under a different user account and is thus independent of other user's resources. Lastly, CloudMan provides support for elastic resource scaling, which can be manual or automatic. The elastic scaling reduces the cost as the certain resources (storage, CPUs, etc) can be removed or added dynamically during the instance runtime, based on the current workload.

CloudMan platform also defines functional but customizable application execution environments (e.g., SMP, MPI, map-reduce). These parallel execution environments are automatically set-up at the instance launch time and made available to users and applications. These environments cover a wide range of uses without requiring users to perform any configuration, which contributes toward making the underlying infrastructure accessible and instantly usable. Simultaneously, because each CloudMan instance of the can be customized, these environments can be adjusted to meet specific application needs. This enabled CloudMan to act as a platform - a platform that allows users to incrementally build on top of an already rich set of preconfigured and functional features. This is in direct contrast with having to recreate all of the lower-level features or mastering an entirely new methodology for working with the cloud infrastructures. This model makes it possible to support the entire life cycle of a tool and an analysis without requiring any special provisions from the developer or the end user.

A. CloudMan arhitecture

CloudMan architecture represents a step toward defining a universal environment for deploying a range of applications on cloud infrastructures. The motivation for this approach is twofold: (1) provide a template for those wishing to migrate their own application to an aggregated infrastructure environment and (2) enable the underlying infrastructure to be interchanged while minimizing changes required to realize the bridge between IaaS and SaaS.

The presented architecture focuses on encapsulating an existing application into a deployable unit that can be instantiated by an individual user. With this approach the applications are abstracted and infrastructure independent. At the same time, a readily deployable application unit allows users to easily and quickly gain access to the desired application. The user is thus not encumbered by having to install and maintain their own infrastructure or installation, nor limited by restrictions imposed by availability of a public deployment of an application. If the application deployment unit exists for multiple infrastructure providers, the user is also free to choose where to run the application. As a result of the self-contained application unit deployment process, the user can independently utilize IaaS infrastructure while enjoying SaaS usability. This approach creates a separation of concerns at a high level and allows everyone involved to focus on their domain instead of being bogged down by management and maintenance routines and requirements.

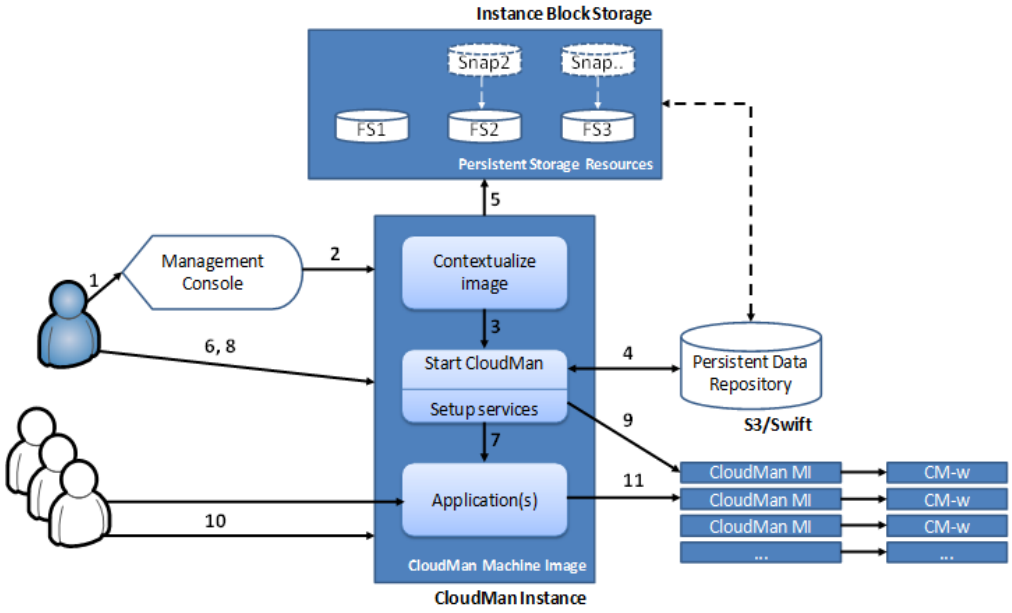

Figure 1. Composition of components composing the arhitecture to enable CloudMan exexution in virtualized environment: (1) The administrator user accesses the infrastructure console manager and requests an instance; (2) an instance is instantiated (this may be an instance with CloudMan already enabled or CloudMan can easily be added to the instance); (3) CloudMan controller starts; (4) the controller contextualizes itself by obtaining needed context from the persistent data repository; (5) any external storage resources are then attached and configured automatically by eCloudMan into appropriate file systems; (6) the administrator user has the option of providing configuration information about their new cluster through eCloudMan's web interface, e.g., cluster size limits; (7) optionally, eCloudMan configures and starts specific application(s); (9) the user can monitor and update deployment parameters via CloudMan, CloudMan automatically scales the cluster, acquiring and releasing additional worker instances as needed but also staying within the predefined constraints; (10) the platform and the application can be used directly or via the eCloudMan API; (11) applications run transparently on the backend resources.

CloudMan is implemented as a standalone web application that acts as a manager for the instantiated deployment. To enable composition of multiple instances into a compute cluster, at the implementation level, CloudMan is represented by two services, a master and a worker. The distinction between services is determined at instance boot time, based on the given instance's role. All of the instance contextualization, master-worker communication, and status reporting is performed through a messaging system implemented using the Advanced Message Queuing Protocol standard and using a RabbitMQ server deployed on the master instance [16].

Conceptually, CloudMan architecture is based on separation and subsequent coordination of otherwise independent infrastructure components: the *machine image*, a *persistent data repository*, and *persistent storage resources* (e.g. data snapshots). The composition and interaction of those three independent components, that comprise the CloudMan architecture, are described in Figure 1. The *machine image* is characterized by simplicity; it consists only of the basic services required to initiate the application unit deployment process. In particular, besides selecting the underlying operating system, only the core set of services is installed on the image itself. Alongside these services, a contextualization script is included in the basic machine image. Contextualization is the process of coordinating the machine instance preparation and deployment at runtime. The contextualization script included in CloudMan's machine image serves only as an access point to the instance while the contextualization details are extracted and stored from the persistent data repository. The *persistent data repository* lives independent of the machine image and is used to provide instance contextualization details, such as, the boot time scripts that define which services should be started. Lastly, *persistent storage resources*, or snapshots, are used as the storage medium for applications, any required tools, libraries, or datasets the application depends on, and the user data.

Once instantiated, these components are aggregated by CloudMan into a cohesive and operational unit with all the features described above becoming functional. As part of the setup, CloudMan automatically configures the master instance as a head node of a Sun Grid Engine (SGE) compute cluster but it does not start any additional worker instances or assign persistent storage to the cluster. In the context of cloud computing, compute instances are usually transient, meaning that any changes made to an instance while the instance is alive are lost at instance termination. In order to persist any data uploaded to the cloud or any analysis results, the data needs to be stored on an external data volume. In the case of CloudMan on EC2, Amazon's Elastic Block Storage (EBS) volumes are used as the persistent storage resources.

At termination time, any read-only persistent storage resources are deleted because they will be recreated from the preexisting snapshots. The persistent storage resources containing user data however are preserved in the user's account. Similarly, all of the cluster configuration settings are preserved in the persistent data repository and will be used upon next cluster instantiation to assemble the complete deployment.

Over the past two years of development, the described architecture has proven very effective and resilient to different cloud providers. Overall, the architecture provides the following benefits:

- Minimum setup time and cost: no need for an external broker

- Automated configuration
- Transparent data persistence
- Self-contained deployment: customizable instances; versioning of tools, data, and configurations

B. Running applications using CloudMan

CloudMan currently supports creation of compute clusters on Amazon's EC2 **Error! Reference source not found.** cloud computing infrastructure as well as OpenStanc and OpenNebula based clouds. The process of instantiating a cluster does not require any computational experience, and requires no compute infrastructure or software beyond the web browser used to control the cluster. Thus, CloudMan is ideal for independent researchers and small labs that have a specific or periodic need for computational resources but lack informatics expertise and commitment to manage and maintain a computational cluster. The process of instantiating a CloudMan compute cluster consists of three steps: (1) obtain an account on the supported cloud (AWS or OpenStack or OpenNebula private clouds), (2) use biocloudcentral.org portal to start a master instance, and (3) use the CloudMan web console on the master instance to manage the cluster size and data persistence. This approach takes very short time to setup a compute cluster and allows a user to have any number of independent clusters, thus supporting easy separation of projects or groups. Once set up, additional users may use the cluster. Depending on the type of application used, user may be able to use the cluster via the web interface or by logging into the instance via command line tools.

For the case of AWS, CloudMan comes preconfigured with more than 100 bioinformatics tools. In addition, at cluster start time, it can be configured to deploy Galaxy application with numerous other tools and 700GB of reference genome data. If a specific tool is not available, it is possible for a user to customize their instance and add the desired tool (see below). Once added, the instance changes may be persisted and shared with other users, as outlined in section B above.

Another interesting example of utilizing CloudMan platform is deployment of the WRF-ARW – weather research and forecasting model. This model requires large storage and computational resources and is favorable to see how it can exploit unique and powerful features of Cloud. In the past, several deployments of the WRF-ARW model has been made on Grid infrastructure [18][19]. All of these attempts of deployments were on a fixed and pre-defined architecture (computing nodes, storage, operating system, etc.).

Regardless of whether is a preinstalled tool or a user-installed one, SGE job manager is configured and used on the cluster for job management. This makes it possible for users to simply copy their job scripts to the cloud cluster and run them there - but with the scalability offered through cloud computing. When a given cluster is no longer needed, the CloudMan web interface can be used to terminate all of the services and worker instances. If persistent data storage was associated with the cluster, the data is preserved while the cluster is offline, and made available in the same state once the cluster is instantiated again. It takes only a few minutes to scale up or down a cluster and consume the required amount of resources.

When installing the set of tools supported by CloudMan platform by default, a script is used to automatically install all the tools. This script is available at https://bitbucket.org/afgane/mi-deployment/ and may be used when customizing individual instances to include the desired tools. Once customized, the user has an option to persist the changes via CloudMan's web interface, thus making those changes readily available for future cluster invocations and/or sharing with other users.

In order to install a custom application, a user instantiates a cloud cluster, modifies the contents of the persistent storage resources (e.g. EBS volumes) by installing the desired tools (performed just like on any other machine), creates a snapshot of the modified data volume (done automatically through CloudMan's web interface), and points their cluster setup to use the newly created data volume snapshot (also done automatically through a web UI). As a result, the functionality and the architecture of CloudMan can easily be adjusted to the computing needs of individual researchers and different domains. This makes it possible for end users to add their own tools to a CloudMan cluster instance while also reusing all of the other provided features.

Note that the implications of such customization mean that one could install a custom tool and provide some sample data. Then persist those changes and share the cluster with the community. Then, users can simply instantiate this shared instance and start using the tool and the data made available without actually having to install or configure anything. Along with the tool, the users were also able to acquire the infrastructure (form the cloud), thus making a given tool instantly accessible to the end users.

C. Application types and the CloudMan platform

Based on the defined architecture of CloudMan, one can define three applications types. Application types are divided depending on where, how and when the applications are installed. These types are:

Type I: a self-contained application that can be installed manually by the user as it would be installed on any user machine or cluster. This application and its dependencies should be installed on the user's persistent data volume and thus it will be retained across invocations of this cluster. The WRF-ARW model is an example for this type.

Type II: an application that wants to be shared with others or may have 3rd party dependencies but none of those include system-level changes. This type of application should be installed on the persistent data repository derived from a snapshot and use CloudMan's features to persist those changes on the otherwise read-only file system. An example for this type of application is SGE.

Type III: an application that requires system-wide install because its dependencies must be a part of the

operating system. For this case, a new machine image needs to be created. Once the new machine image is created, future invocations of this cluster should use that particular machine image. GalaxyTool is an example of this type.

III. CONLUSION

To keep up with the growth of research data being produced and the accompanying computational demand required to process those data, there is a need for increased access to computational resources. Cloud computing offers access to such resources but still makes it difficult to create complex deployments of useful standalone infrastructures. This is especially cumbersome for individuals and small labs that lack informatics support to fully harness this general-purpose infrastructure.

This paper showcases how CloudMan can be used to lower the barrier of entry into cloud computing while readily leveraging many of the features of the cloud. Namely, CloudMan handles all of the intricacies of cloud computing resource acquisition, configuration, and scaling to deliver a personal compute cluster in a matter of minutes. All of the interaction with CloudMan and the associated cloud cluster management is performed through a web based user interface and requires no computational expertise. Alternatively, if advanced access to the system is desired, one can easily gain access while continuing to operate in a traditional cluster environment.

Overall, the CloudMan platform can be preconfigured with numerous applications that are ready to be used. The installation procedure for those tools is fully automated thus supporting reproducibility. In addition, three different cloud application types were categorized according to the way of their deployment as potential candidates for customizing a given instance of CloudMan.

ACKNOWLEDGMENT

The authors acknowledge the support of scientific research project “Methods of scientific visualization” (098-098 2562-2567), founded by the Ministry of Science, Education and Sports of the Republic of Croatia. Additionally, the authors would also like to thank the Galaxy Team who was responsible for deployment of the Galaxy application with CloudMan.

REFERENCES

[1] Afgan E., Baker D., Coraor N., Goto H., Paul I.M, Makova K.D., Nekrutenko A., Taylor J., "Harnessing cloud computing with Galaxy Cloud," Nature Biotechnology, Vol 29, Issue 11, 2011.

[2] Christian Vecchiola, Suraj Pandey, and Rajkumar Buyya, “High-Performance Cloud Computing: A View of Scientific Applications, in Proceedings of the 10th International Symposium on Pervasive Systems Algorithms and Networks, 2009, p. 13

[3] Above the Clouds: A Berkeley View of Cloud Computing

[4] Zhang, Q., Cheng, L., Boutaba, R.: Cloud computing: state of the art and research challenges. Journal of In-ternet Services and Applications 1(1) (2010)

[5] A. Menychtas, G. Kousiouris, D. Kyriazis, and T. Varvarigou, “Minimizing technical complexities in emerging cloud computing platforms”, in Euro-Par 2010 Proceedings of the 2010 conference on Parallel processing, 2010, pp. 603-610.

[6] “Mosaic Cloud.”, http://www.mosaic-cloud.eu/

[7] “4CaaSt.”, http://4caast.morfeo-project.org/

[8] “Contrail.”, http://contrail-project.eu/

[9] “CumuloNimbo.” , http://cumulonimbo.eu/

[10] “StratusLab.”, http://stratuslab.eu/doku.php/start

[11] “StarCluster.”, http://web.mit.edu/stardev/cluster/

[12] “Red Hat OpenShift.”, https://openshift.redhat.com/app/

[13] “Manjrasoft Aneka.”, http://www.manjrasoft.com/products.html

[14] “Cirrocumulus.”, http://knoesis.org/research/srl/projects/cirrocumulus/.

[15] E. Afgan, D. Baker, N. Coraor, B. Chapman, A. Nekrutenko, and J. Taylor, “Galaxy CloudMan: delivering cloud compute clusters.”, BMC bioinformatics, vol. 11 Suppl 1, no. 12, p. S4, Jan. 2010.

[16] Afgan E., Baker D., Taylor J., “A Reference Model for Deploying Applications in Virtualized Environments,” Concurrency and Computation: Practice and Experience, 2011.

[17] Afgan E., Goecks J., Baker D., Coraor N., the Galaxy Team, Nekrutenko A., and Taylor J., "Galaxy - a Gateway to Tools in e-Science," in Guide to e-Science: Next Generation Scientific Research and Discovery , K. Yang, Ed., ed: Springer, 2011, p. 145-177.

[18] Davidović, D.; Skala, K.; Belušić, D.; Telišman-Prtenjak, M., “Grid implementation of the Weather Research and Forecasting model,” Earth Science Informatics, vol. 3, issue 4, pp.199-208, 2010

[19] Davidović, Davor; Skala, Karolj. Implementation of the WRF-ARW prognostic model on the Grid, Proceedings Vol. I. MEET&GVS 33rd International Convention MIPRO, pp. 253-258, 2010