A Comprehensive Solution for Research-Oriented Cloud Computing

Mevlut A. Demir(✉), Weslyn Wagner, Divyaansh Dandona, and John J. Prevost

The University of Texas at San Antonio,
One UTSA Circle, San Antonio, TX 78249, USA
{mevlut.demir,jeff.prevost}@utsa.edu, {weslyn.wagner,ubm700}@my.utsa.edu

Abstract. Cutting edge research today requires researchers to perform computationally intensive calculations and/or create models and simulations using large sums of data in order to reach research-backed conclusions. As datasets, models, and calculations increase in size and scope they present a computational and analytical challenge to the researcher. Advances in cloud computing and the emergence of big data analytic tools are ideal to aid the researcher in tackling this challenge. Although researchers have been using cloud-based software services to propel their research, many institutions have not considered harnessing the Infrastructure as a Service model. The reluctance to adopt Infrastructure as a Service in academia can be attributed to many researchers lacking the high degree of technical experience needed to design, procure, and manage custom cloud-based infrastructure. In this paper, we propose a comprehensive solution consisting of a fully independent cloud automation framework and a modular data analytics platform which will allow researchers to create and utilize domain specific cloud solutions irrespective of their technical knowledge, reducing the overall effort and time required to complete research.

Keywords: Automation · Cloud computing · HPC
Scientific computing · SolaaS

1 Introduction

The cloud is an ideal data processing platform because of its modularity, efficiency, and many possible configurations. Nodes in the cloud can be dynamically spawned, configured, modified, and destroyed as needed by the application using pre-defined application programming interfaces (API). Furthermore, cloud-based systems are highly scalable and can be configured to provide a high degree of availability [1]. For these reasons, industry has fully embraced cloud computing for delivering services and applications, as well as establishing dedicated jobs focused on the maintenance and management of cloud infrastructure. For all the reasons stated, a cloud-based solution would also be optimal for academic

M. Luo and L.-J. Zhang (Eds.): CLOUD 2018, LNCS 10967, pp. 407–417, 2018.
https://doi.org/10.1007/978-3-319-94295-7_29

research [3]. Currently, most researchers utilize local compute systems to run calculations and store required data. These calculations are often complex and can take multiple hours to days to converge to a conclusion. A cloud-scaled solution could outperform local compute systems, while also allowing the researcher to quickly change the underlying topology in response to transient parameters or characteristics of the models used.

In academia, however, Infrastructure as a Service (IaaS) solutions have not been adopted for two main reasons: lack of technical expertise and the need for dedicated management. IaaS requires the user to fully design and configure their cloud environment which may consist of runtime, middle-ware, and operating systems [7]. This configuration must then be maintained so that software can run efficiently on top of it. Researchers, who are experts in their own field of research, usually lack the technical skills to create and manage a cloud solution, and become reliant upon their institution's IT department to provide them with assistance. Those who desire to possess the requisite skills are forced to divert time from their research to learn the necessary skills themselves. Cloud providers recognize this issue and have created systems such as one-click installs or automation frameworks to greatly reduce the effort and expertise needed to create cloud infrastructure, but these solutions are not holistic in nature nor geared towards academia.

To address this problem, we propose Research Automation for Infrastructure and Software framework (RAINS). RAINS aims to be an all-in-one infrastructure procurement, configuration, and management framework tailored specifically towards the needs of researchers [2]. The framework is agnostic to the cloud provider giving the researcher the option of hosting with different public providers, or running a private cloud on premises.

RAINS incorporates a crowd-sourced model to allow researchers to share and collaborate their unique infrastructure builds across the world in a seamless manner. This would enable researchers to quickly reproduce the results of ongoing research, and share optimum configurations.

RAINS will make the IaaS model accessible to researchers in a manner that is easy to use and configure, thus allowing them to focus solely on their research and not on extraneous system configuration.

Once a custom infrastructure and software environment has been created, software deployments will need to be connected and synchronized before applications are fully operational in the cloud. There are hundreds of active, open source tools for big data analytics - each one has a custom installation, working environment, and communication method. For a typical researcher, learning all of these tools and their selective APIs can be a massive hurdle. These big data tools often work in a clustered topology consisting of a data ingestion layer, data processing layer, data storage layer, and a data visualization layer.

This further adds complexity to utilizing IaaS because it is not directly apparent which set of tools would be correct for the researcher and what type of environment and dependencies each one has. A researcher would have to have

advanced knowledge of programming, data science, and system administration to connect these individual tools into a cluster formation and run them successfully.

To speed up integration we propose Software Platform for Experimental Analysis and Research (SPEAR). SPEAR is designed to allow for easy integration between various big data and machine learning tools to create a working research cluster atop a RAINS-created infrastructure. The platform takes advantage of the cloud's high availability and scalability to handle numerous concurrent data streams for parallel processing and real time visualization. SPEAR empowers the end user to prototype and execute data processing algorithms in an abstracted environment by independently handling data migration. SPEAR offers plug and play support for popular data acquisition, control, or analysis tools that the end user may wish to integrate into their custom installations.

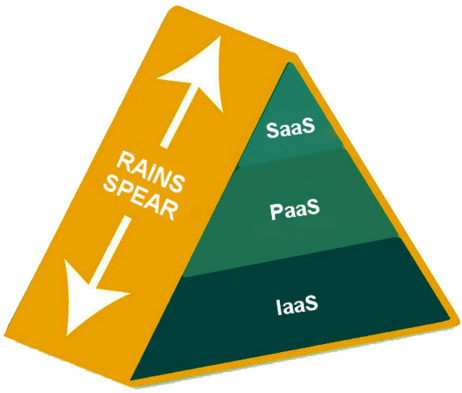

Fig. 1. SPEAR and RAINS integrate IaaS, PaaS, and SaaS

The emergence of cloud-based services in the early 2000's revolutionized Internet-based content delivery. Today, almost every interaction we have with an on-line service is being handled through a cloud-based system [4]. Cloud services can be classified as either Software as a Service (SaaS) [6], Platform as a Service (PaaS) [5], or Infrastructure as a Service. Under, SaaS a user can use the software without having to worry about its underlying hardware, software environment, or session which are all being managed be the service provider. PaaS allows a user to interact with the cloud service at an operating system level. IaaS gives complete control over the underlying virtual or physical hardware stack, to the end user, allowing for a truly complete customizable experience. Together RAINS and SPEAR provide the researcher with solution as a service which integrates IaaS, PaaS, and SaaS as shown in Fig. 1.

2 RAINS Framework

The aim of RAINS is to provide easy-to-use cloud automation methods that can be used standalone, or as a base for software platforms such as SPEAR. RAINS has a modular design in which the back-end and front-end applications are coded separately and communicate through established API calls. The front end is designed to run the browser and allow for ease of access. The back-end consists of a REST API which interfaces with the orchestration engine to procure and deploy the designed infrastructure.

2.1 Dashboard

The RAINS dashboard is a browser-based graphical user interface intended for configuring, designing, managing, and deploying cloud infrastructure. The dashboard consists of a palette and a canvas for visualization. RAINS offers three viewing modes: Machine Level, Application Level, and Network Level. Each view has a custom palette and canvas associated for managing aspects of the infrastructure build. The framework employs a drag-and-drop API in conjunction with form-based wizards to create representational states which can then be rendered to the canvas. Upon initialization the user can drag and drop hardware and software components to create their unique cloud infrastructure build. Once the infrastructure build has been graphically designed, the deploy button realizes the infrastructure.

Machine View. The Machine View focuses on the machine level aspects of the infrastructure. The palette provides virtual machines and containers as draggable components. The canvas renders machine level details such as hostname, IP address, and machine type.

Application View. The Application View focuses on the application and software aspects of the infrastructure. The palette offers a list of supported software as draggable components. The canvas renders application level details such as software name and current status.

Network View. The Network View highlights TCP/IP network configurations for each machine nodes in detail, as well specific ports that applications are attached to.

Dragged and dropped components on the canvas are referred to as droplets. A droplet consists of a machine level component and may also have software and networking components attached. RAINS makes use of forms, models, and templates to prompt the user for basic information needed to satisfy a particular droplet's requirements. In case the user cannot provide basic information, the user is given the choice to use default options or pull a build from trusted repositories. The combination of all the droplets on the canvas represents the

complete infrastructure build the user wishes to deploy. Once an infrastructure build is deployed, the canvas continues to serve as a real-time representation of the active build and can be used to dynamically or manually reconfigure the infrastructure. Each droplet host key-value pairs which describe the infrastructure build, its own canvas styling, and the droplet's current status if it is actively deployed. The revolutionary novelty of RAINS is that infrastructure builds can be further packed into droplet clusters with exposed endpoints, allowing for integration of two complex cloud systems comprised of many droplets. In this manner, systems can increase in complexity while still remaining easy to understand through encapsulation and nesting.

2.2 Canonical Form

When a user builds an infrastructure set, RAINS dynamically describes the build in a JSON-based form. This format encapsulates all droplets and droplet clusters into a stateful text based representation. The JSON format can be shared with others or saved to create a snapshot of the current infrastructure build. This representation is then passed to the back end which uses the JSON format to generate a complete canonical form. This form consists of a human readable task list for orchestration and procurement in YAML format. Using a single canonical form facilitates repeatability and completely decouples the orchestration file from the task list. A text-based markup or object level format is easier to share between peers and is computer-friendly enough for quick parsing, leading to a higher degree of integration with other available systems that understand markup. The canonical form can then undergo translation and mapping to create orchestration files specifically tailored for each major cloud platform including AWS, Google Cloud, and OpenStack.

2.3 REST API

RAINS uses the traditional HTTP-based REST API [13] and database combination to provide service communication. The REST API allows the front-end dashboard to access the database to store session information, get updates, and post JSON forms for translation. RAINS utilizes both a relational and non-relational databases to store data. The REST API can also be used internally to serve as a communication broker between server processes.

2.4 Synthesis

The JSON output from the front end is the most minimal description of the infrastructure build. When the JSON form is POSTed to the backend, hopper scripts parse and sort the information based on whether it is a procurement or orchestration directive. After sorting, a task-based canonical form is generated, which consists of a list of the procurement and orchestration tasks needed to realize the infrastructure. The canonical form gives precedence to procurement

directives while orchestration directives are listed afterwards. The YAML-based canonical form can then be translated into orchestration templates and Ansible [8] execution Playbooks.

2.5 Translation

The canonical form alone is not an inclusive enough input for an orchestration engine. The key-value pairs in the canonical form need to be extended into orchestration scripts that the orchestration engines can use. A translation service uses advanced templating and mapping to generate the vendor specific consumables for AWS [15], Google Cloud Platform [10], OpenStack [14], Azure [11], and Ansible. The current implementation of the translation layer targets AWS and OpenStack Heat, while also supporting Ansible for pre- and post-deployment orchestration. RAINS provides AWS Cloud Formation Templates and OpenStack Heat Orchestration Templates [9], and allows for automatic procurement, scaling, and other tasks supported by the providers' respective APIs. RAINS also supports running user-provided Ansible scripts against nodes running in an infrastructure build. The user can provide Ansible scripts via the dashboard and then execute them on one or multiple droplets that are active on the canvas.

2.6 Procurement

Once the orchestration scripts are ready, RAINS interfaces with the respected APIs of the cloud provider the user has chosen. The user's account details and subscription plan are taken into account, and an API call is made to procure the machine level nodes, and prepare them for Ansible. Ansible is then utilized to complete the environment creation and software installation tasks. At this point RAINS has created the infrastructure in the cloud.

2.7 Real-Time Feedback Loop

A real-time feedback loop is currently under development for RAINS. The feedback loop will connect to the cloud platforms' metrics and logging services, such as Ceilometer for OpenStack [12]. These metrics will map to layout styling and will be reflected onto the canvas for visualization. The real-time feedback loop will allow RAINS to check the status of the computational nodes and the software stacks running on them in real-time. RAINS will then be in the position to manage the health of nodes, dynamically scale applications up, down, or out depending on load, and even make optimization suggestions to the user on how to best implement an architecture build. Active monitoring will also reduce computational waste and help the user reduce deployment costs.

2.8 Repository

To help foster a collaborative environment and provide for easy templating, a dedicated repository will be used for canonical form tracking and storage.

Researchers will have the option of initiating official builds and custom wikis to highlight the usage and performance of their infrastructure builds. The online repository will serve as a sharing and storage portal through which researchers can pull, push, and share their unique configurations. A voting-based ranking system will be utilized to give credibility to builds.

3 SPEAR

The SPEAR aims to give researchers access to open source big data analytics tools as well as popular machine learning, robotics, and IOT frameworks in a easy to use and integrated environment. SPEAR at it's core is a plug and play platform manager. It manages the integration of various different suites of software into one cohesive clustered application. Furthermore, the usage of the cloud allows for hot swapping of applications in real time, resulting in a robust and resilient system. SPEAR can easily be layered atop RAINS, and employed to connect node endpoints generated by RAINS, and even program and run applications. A typical SPEAR cluster will consist of a data ingestion layer, data processing layer, data storage layer, and a visualization layer, while providing for additional application specific layers such as machine learning and robot operating system (ROS) [16] layers.

Data Ingestion. The data ingestion layer consists of a distributed queuing system along with a message broker. Data ingestion in this manner can scale effectively across cloud nodes to handle multiple concurrent data streams.

Data Processing. The data processing layer consists of big data analytical tools such as distributed stream processing engines and batch processing engines. Distributed stream processing engines are ideal for real time stream based processing. Batch processing can be performed after data has been collected on large aggregates of data.

Data Storage. The data sink layer consists of storage volumes and databases. Users can configure SQL relational databases or NoSQL non-relational databases to serve as data sinks for post processing or direct storage. SPEAR takes into account the effort needed to create database tables and schemas, and uses custom graphical tools to help the user complete this task to reduce the need for a user to learn CLI tools.

Visualization. The visualization layer can consist of various different output forms. HTML5 and Javascript based visualization techniques are very popular, but data can be fed into graphing and mapping programs as well.

3.1 Graphical User Interface

SPEAR's graphical user interface employs graphical wires to define cluster topologies. Users can click and drag wire connections to and from component endpoints to set TCP/IP communication across a cluster. An emphasis on block programming is given to abstract away the necessity for configuring and using each component of the big data architecture. Graphical Programming Interfaces can be utilized to abstract the need to learn specific tools and coding techniques. Instead the end user can click and drag functionality or use graphical wizards to modify, customize, and utilize the underlying software applications.

3.2 Topology Creation

RAINS can create infrastructure and software environments, but it is not designed as a software management platform. SPEAR takes on this responsibility by connecting the individual nodes into the formation of a cluster. SPEAR does this by utilizing various communication tools such as TCP/IP port mappings, communication scripts, and pub/sub message protocols.

3.3 Data Flow

To facilitate the flow of data between ingestion, processing nodes, and visualization SPEAR currently supports Kafka [17], RabbitMQ [18], and Apache Pulsar [19]. Internally publish/subscribe models are utilized for inter-cluster communication. In a typical pub/sub configuration, data packets travel through queues. Each queue consists of a bidirectional channel which supports quality of service level 1. As the load on the system changes, the queues can easily be scaled and parallelized to sustain a high throughput.

3.4 Processing

SPEAR supports popular distributed stream processing engines as well as batch processing engines to satisfy processing needs. Distributed stream processing engines can interface directly with most messaging queues to ingest data and perform permutations and computations. Popular DSPEs such as Apache Storm [20] and Apache Heron [21] utilize controller and worker nodes to create a highly scalable, fault tolerant processing solution. Apache Spark [22] and Hadoop [23] are popular batch processing platforms that can easily be integrated into a SPEAR cluster.

3.5 Data Sink

Data must be persisted in some form for later analysis and reference. SPEAR supports relational databases like MySQL [24], and non relational databases such as MongoDB [25]. Some applications such as Hadoop require custom NoSQL implementations like HBase [26]. SPEAR's graphical interfacing tools will allow users to perform create, update, read, and delete data from the database.

3.6 Visualization

SPEAR supports browser based visualization and allows for easy integration with JavaScript visualization frameworks such as D3 [27]. JavaScript and HTML5 have become popular for visualization because of universal access via browsers, and low maintenance costs. SPEAR offers visualization templates using D3, from which a user can pull data from databases or from data processing engines for real time or aggregate visualization, without having to create the visualization mechanisms themselves. An example framework can be seen in Fig. 2.

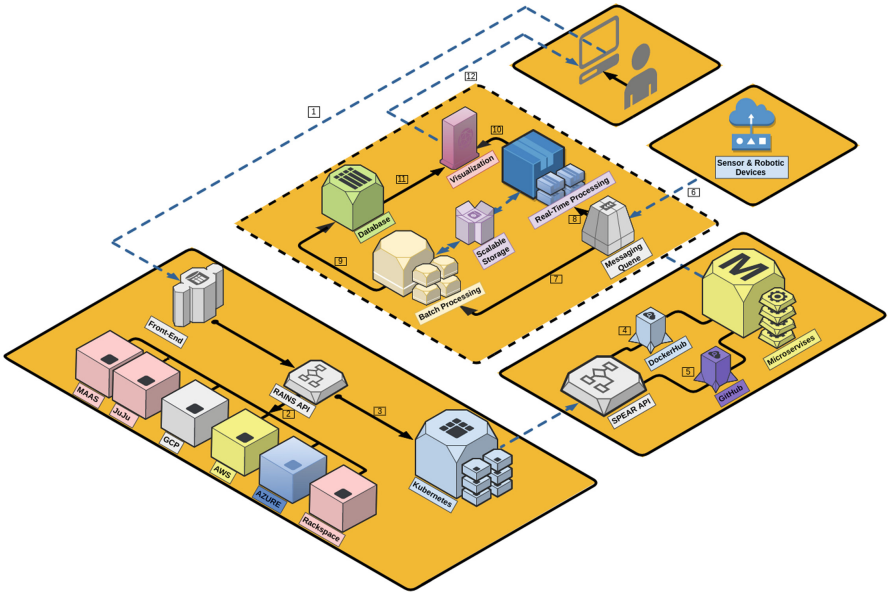

Fig. 2. Overview of the SPEAR & RAINS frameworks

4 Future Work

We plan to utilize RAINS and SPEAR in our research laboratory as a test case. In this manner we can add functionality and manage the development cycle of each software based on the needs of university researchers. RAINS and SPEAR will be utilized in our cyberphysical IOT systems research. We plan to gain performance benchmarks and metrics and further improve our software.

5 Conclusion

The layering of SPEAR and RAINS creates a holistic cloud-based solution for procuring cloud infrastructure and data processing that is customizable and

controllable from end to end, that simultaneously mitigates the need for technical cloud expertise from the researcher. RAINS can serve as the starting base for other platforms which may require a highly customizable cloud-based infrastructure, or can be used standalone to solely generate cloud infrastructure. The SPEAR platform will give researchers quick and seamless access to many current data analytics tool sets without the need to learn complicated coding and command line tools. Together, RAINS and SPEAR will revolutionize research speed and give the researcher full access to the cloud's potential.

References

1. Kondo, D., et al.: Cost-benefit analysis of cloud computing versus desktop grids. In: 2009 IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2009. IEEE (2009)
2. Wagner, W.S.: Research automation for infrastructure and software: a framework for domain specific research with cloud computing. Dissertation. The University of Texas at San Antonio (2017)
3. Benson, J.O., Prevost, J.J., Rad, P.: Survey of automated software deployment for computational and engineering research. In: 2016 Annual IEEE Systems Conference (SysCon). IEEE (2016)
4. Qian, L., Luo, Z., Du, Y., Guo, L.: Cloud computing: an overview. In: Jaatun, M.G., Zhao, G., Rong, C. (eds.) CloudCom 2009. LNCS, vol. 5931, pp. 626–631. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10665-1_63
5. Malfara, D.: Platform as a service. Working paper ACC 626, University of Waterloo, Waterloo, Canada, 3 July 2013
6. Gupta, N., Varshapriya, J.: Software as a service. Int. J. Innov. Res. Adv. Eng. (IJIRAE)-2014 **1**(6), 107–112 (2014)
7. Manvi, S., Shyam, G.: Resource management for Infrastructure as a Service (IaaS) in cloud computing: a survey. J. Netw. Comput. Appl. **41**, 424–440 (2014)
8. Ansible: How Ansible Works (2018). https://www.ansible.com/how-ansible-works
9. OpenStack: Heat-Translator (2018). https://wiki.openstack.org/wiki/Heat-Translator
10. Google Cloud Platform: About the Google Cloud Platform Services (2018). https://cloud.google.com/docs/overview/cloud-platform-services
11. Microsoft Azure: Azure regions, more than any cloud provider (2018). https://azure.microsoft.com/en-us/
12. OpenStack: OpenStack Docs: Welcome to Ceilometer's Documentation (2018). https://docs.openstack.org/ceilometer/latest/
13. Fielding, R.T., Taylor, R.N.: Architectural styles and the design of network-based software architectures, vol. 7. Doctoral dissertation, University of California, Irvine (2000)
14. OpenStack: OpenStack Docs: System Architecture (2018). https://docs.openstack.org/
15. AWS (2018). https://aws.amazon.com/
16. ROS (2018). http://www.ros.org/
17. Kafka (2018). https://kafka.apache.org/
18. RabbitMQ (2018). https://www.rabbitmq.com/
19. Apache Pulsar (2018). https://pulsar.incubator.apache.org/

20. Apache Storm (2018). http://storm.apache.org/
21. Apache Heron (2018). https://twitter.github.io/heron/
22. Apache Spark (2018). https://spark.apache.org/
23. Hadoop (2018). http://hadoop.apache.org/
24. MySQL (2018). https://www.mysql.com/
25. MangoDB (2018). https://www.mongodb.com/
26. HBase (2018). https://hbase.apache.org/
27. D3 (2018). https://d3js.org/