Journal of Engineering Science and Technology Review
Special Issue on Telecommunications, Informatics, Energy and Management 2019

JOURNAL OF
Engineering Science and Technology Review

Conference Article

www.jestr.org

Scaling and Automation in Cloud Deployments of Enterprise Applications

Strahil Sokolov, Orhan Idiriz, Mihail Vukadinoff and Stefan Vlaev

Department of information Technology, University of Telecommunications and Post, 1 Acad. St. Mladenov Str., 1700 Sofia, Bulgaria

Received 30 September 2019; Accepted 21 February 2020

Abstract

An approach is proposed for utilizing modern tools and algorithms for scaling and automation of cloud deployments of Enterprise Applications. Modern cloud based solutions for the enterprises have become even more popular than traditional infrastructure deployments. A great number of open-source and commercial automation and scaling solutions are available. There are challenges in terms of performance, ease of use and sustainability of automated application provisioning in the cloud. In previous research we presented targeted criteria for choosing a suitable provider of cloud storage and solution for scaling cloud-based solutions with thousands of users. Our goal is to establish a set of criteria when choosing a suitable scaling and automation approach for enterprise applications.

Keywords: cloud deployment; automation, auto-scaling.

1. Introduction

Cloud application deployments increase the need for performance efficiency and security. They are the most active research fields among the worldwide cloud community. These terms already are regarded together in the QoS for cloud providers [1, 2]. Furthermore, these terms are due to the latest trend in the IT industry: "transformation". It requires business to change the model of how they develop new applications, how they deploy them, how they support them and even how they store the end user data. Faster access to the information becomes crucial in the era of cloud computing and fast mobile networks.

Scalability and performance seem to be the most attractive features of the modern cloud infrastructure of the popular providers. Solutions for deployment on cloud infrastructure should be able to scale and be open for optimization.

Various methodologies exist for scaling cloud application deployments based on performance estimation via network measurements such as network latency. In our current work we propose criteria for designing scalable environments for cloud deployments of enterprise applications.

The survey [3] describes in accurate manner the layers of the centralized cloud computing model with the following general layers:

- Layer (I) Centralized cloud computing layer: contains the provider cloud datacenters. This layer is used for long-term storage and application-level data processing operations that are typically less time-sensitive. Applications in this layer may have different, service modules, each one with a different purpose for high-level data processing according to users' requirements.
- Layer (II) SDN/NFV technologies layer: Software-Defined Networking (SDN) and Network Functions Virtualization (NFV) are contemporary trends for creating innovative network service from design to implementation and operations. They support data transition between edge nodes and cloud datacenters.
- Layer (III) Edge computing layer: The edge computing layer represents gateways and data collection services acting on raw data-aggregating, filtering, encrypting and encoding the local data streams online. This layer is where the cloud resources are being distributed and moved near to the end-users and end-devices. Edge computing has often been called ubiquitous computing as well. This layer can help reduce traffic from the core network and datacenters.

The rest of the paper is organized as follows: In section 2 features of the cloud are described for choosing performance metrics. In section 3 the scaling criteria are described. Section 4 gives experimental results.

2. Material and method

2.1 Concept for scalable enterprise applications in the cloud

The concept [4] of the cloud is that it is designed to provide conceptually infinite scalability. One can leverage and benefit from the cloud architecture only when the application architecture is scalable. Therefore, monolithic

*E-mail address: 1strahil.sokolov@gmail.com
ISSN: 1791-2377

components and bottlenecks in the existing architecture have to be detected; areas where the on-demand provisioning capabilities in the architecture cannot be leveraged have to be identified; the application has to be refactored in order to leverage the scalable infrastructure and take advantage of the cloud. Building scalable applications is presented in terms of the most significant features of a scalable application:

- Resource increase would result in a proportional increase of performance
- Scalable services are capable of handling heterogeneity
- Scalable services are operationally efficient
- Scalable services are resilient
- Scalable services should become more cost effective with their growth (Cost per unit is reduced as the number of units increases).

2.2 Monitoring contexts and performance metrics

This section describes the objects from the different layers of the centralized cloud model and the respective metrics.

• VM-level metrics

Scaling the enterprise cloud application can be done through analyzing the utilization of the VM resources – CPU, Memory, Disk. If the CPU utilization nears 100% and the CPU run queues are close to becoming full, the system runs out of available processing capacity - action must be taken to adapt to the new challenge – or, preferably, before that point, in predictive analysis. Memory usage shows the percentage of memory that is used on the selected machine. If the memory consumption is too high, again adaptation has to take place. Disk usage has to do with the amount of data read or written by a particular VM. Disk usage is also an indication of the percentage of used drive space. Allocating more storage to the VM and allocating it to the appropriate storage area can resolve disk space issues on many occasions. Network usage represents the traffic volume on a particular VM interface, including external and internal data traffic.

2.3 Scaling enterprise applications in the cloud

On Fig 1. the general workflow of the scaling approach is presented. The performance metrics are gathered from the respective application instances. The scaling thresholds are adapted based on the customer requests according to (2). The performance is evaluated based on the adapted thresholds, then Scaling is executed to provide more instance and improve performance or to reduce the number of instances automatically. If there is no need for scaling of the environment, the current state is kept.

An example [2] is provided from a conducted performance estimation and tuning of a relatively large automated Amazon Web Services (AWS) environment. This environment hosts the server side of a researched mobile application for a e-learning provider that has video-streaming instructor-lead trainings and provides means to students for voting on questions during an online training. With major lecture and quiz being held only once per week, it's would fit the use case for a cloud on-demand environment, that is only being deployed for the training, and all virtual machines being shutdown or destroyed at the end. This helps cut costs to a minimum, and eliminating the usage of a private data center for the purpose. The load comes from the large set of online users responding to certain challenge within a limited interval. Such scenario is usually a challenge to any online system.

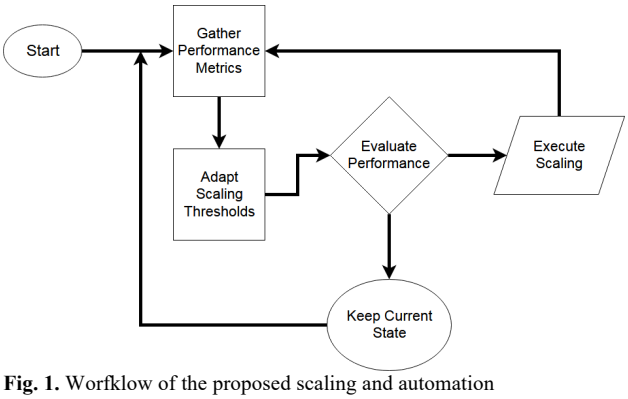

Fig. 1. Worfklow of the proposed scaling and automation

By using Chef functionality and AWS OpsWorks the online learning virtual landcape is deployed within minutes, shortly before the event. Advanced automation is used to make the environment scalable with the number of virtual machines being adjusted based on the expected number of requests. 15 instances of HAProxy are available in advance, 30 instances of web servers running applications based on Django. In addition, there are 12 cache servers based on Memcached which are also using AWS Aurora and DynamoDB for data storage and also ElasticCache service of Amazon Web Services with Redis. There are certain capabilities that can be carried out asynchronously and for this RabbitMQ and Celery is being used. Media files and static content are stored on the Amazon S3. A set of tools has been evaluated for operating and automating the deployment of the infrastructure. Below the environment components (as shown on Fig. 2) and the evaluated tools in this research are given in more detail.

A. HAProxy

HAProxy - a free and open source software which is able to provide high availability load balancer and proxy server for TCP and HTTP-based applications. It is able to scatter requests across multiple backend servers, it is written in C and is famous for being fast and efficient.

B. Django

Django - a free, open-source web framework, written in Python, which follows the model-view-template architectural pattern. The Django Software Foundation is an independent organization established as a 501 non-profit and provides support and development for this framework.

C. Puppet

Puppet [5] is an open source server automation tool that helps automate the configuration and management of the IT infrastructure with modern. Puppet is configured within an agent-master architectural platform, where a master node would hold all configuration information for a multitude of managed configuration items (nodes) running an agent. Its main features include fast resource discovery; easy provision of new nodes in cloud, hybrid or physical environments; configuration of setup ranges; orchestrating changes and events across clusters of nodes.

D. Chef

Chef [6] is a powerful automation tool and infrastructure as a code language. It operates within a client-server scheme. An agent is called Chef Client and runs on each managed

node. It connects to a Chef Server in periodically connecting to a Chef Server to download and evaluate configuration code, known as recipes. If no changes are necessary, there would be no modifications to the system.

E. Ansible

Red Hat® Ansible® [7] is an IT automation tool that transforms repetitive, inefficient tasks of software release cycles into predictable, scalable, and simple processes. Ansible by Red Hat is able to provide automation for configuration management, application deployment, cloud provisioning and service orchestration.

F. Terraform

HashiCorp Terraform [8] is an automation framework that is oriented towards creating, changing and improving IT infrastructure. It is free and open source and transforms APIs into configuration files that can be shared amongst team members, treated as code, edited, reviewed, and versioned.
It allows reusing infrastructure design and deployment on various cloud providers.

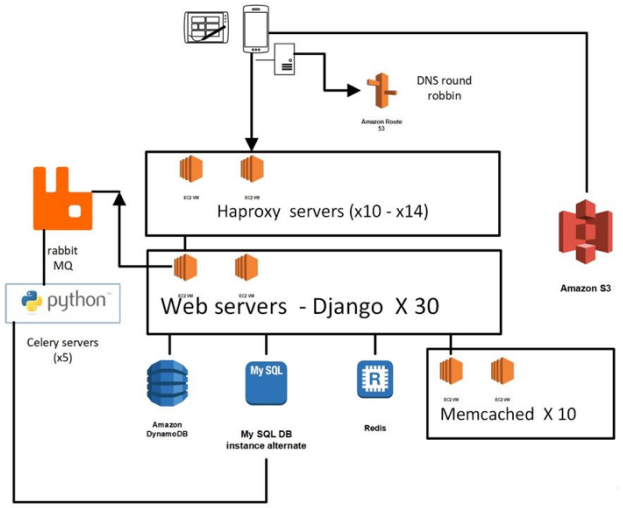

Fig. 2. Environment architecture

Chef configuration management is used for the dynamic evaluatoin of the backend limits:

$$N_{web-server-instances} / N_{HAProxy-instances} \quad (1)$$

Increasing the number of HAProxy loadbalancer instances becomes a drawback: this is due to the fact that maximum connections per backend are being reduced. The scaling algorithm would assume that if the connections are properly balanced, then the total of all backend limits from the HAProxy instances will increase. Due to the uneven AWS DNS RR the environment can handle a single HAProxy instance at a time. The risk is that that the backend limit per load balancer could be hit.

Load tests were conducted a different type of record was used: a single A record holds all the destination IPs and then the client would select one and connect to it. This behaviour was not reproduced during the tests. According to AWS documentation there is a limitation for up to 8 IPs per A-Record.

2.3.1 Adapting the scaling threshold

The scaling threshold (1) is a dynamic measure which is being derived from the current state of the environment. The adaptation of the coefficient is via KAMA.

3. Results and discussion

The screenshots on Fig. 3 show session snapshot data from two different HAProxy load balancers at the same interval of time. The current number of sessions is depicted in Sessions Cur (current count) where unevenly distributed connections are observed between the HAProxy instances.

On Fig. 3. are shown the metrics from a server with few open sessions.

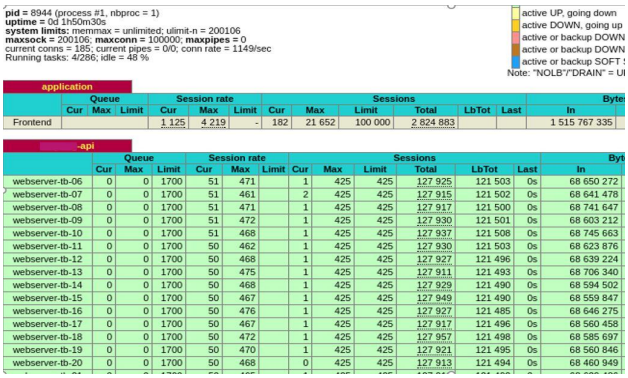

Fig. 3. Distribution of sessions among the HAProxy instances (lower-to-no load).

The graph over time in the Datadog monitoring is also showing a hint towards this. Every color is a single HAProxy session count. It appears that using this procedure we were able to detect that AWS Route 53 is balancing in specific way in which it is sending the first proxy IP to *all* the clients then after a few seconds sends the second IP to all clients. In this way, all clients end up connecting to the same HAProxy for a certain time-frame. They are being balanced over time, but peaks are piling up only on a single HAProxy which is prone to be overloaded.

Fig. 4. Performance coefficient calculated for all balancer servers over time.

4. Conclusions

Our ongoing experiments delivered the following outcomes:

- Scaling of the HAProxy instances during an online streaming event to a maximum of 8, suggesting four. The benefit from this would be that even if customers end up connecting to the same HAProxy balancer, the per backend, per proxy limits will be higher. The data shows that the proxy machines themselves are heavily loaded and can handle significantly large number of requests. A signle limitation represents the active tcp connection limit of

around 65000 tcp sessions per HAProxy. This would mean that 4 instances should easily cover 200K open connections.

- Single DNS entry should be used with multivalue answers. Better distribution of requests has been proved with load tests.
- Scaling efficiency can be achieved through using automation tools and implementing the suggested approach for autoscaling.

In the future this work will be extended into extending the scaling approach to hybrid cloud and improving the adaptive threshold calculation by means of neural network.

Acknowledgements

This work is supported by the University of Telecommunications and Post (UTP), Sofia, Bulgaria, internal research grant Nr NID16/03.04.2018 - "UNICLOUD2.0: Development and integration of cloud services in the learning process of UTP".

References

1. Khazaei, Hamzeh, Jelena Misic, and Vojislav B. Misic. "Performance analysis of cloud computing centers using m/g/m/m+ r queuing systems." IEEE Transactions on parallel and distributed systems 23, no. 5 (2012): 936-943.
2. Sokolov, Strahil A., Stefan M. Vlaev, Mihail Vukadinoff, Asen P. Zahariev, Teodor B. Iliev, and Ivaylo S. Stoyanov. "Performance Estimation of Scalable e-Learning Systems in the Cloud." In 2018 IEEE 24th International Symposium for Design and Technology in Electronic Packaging(SIITME), pp. 148-151. IEEE, 2018.
3. Marschall, Matthias. Chef infrastructure automation cookbook. Packt Publishing Ltd, 2015.
4. Varia, J., 2010. Architecting for the cloud: Best practices.
5. Loope, James. Managing infrastructure with puppet: configuration management at scale. " O'Reilly Media, Inc.", 2011.
6. Taylor, M. and Vargo, S., 2014. “Learning Chef: A Guide to Configuration Management and Automation." O'Reilly Media, Inc..
7. Geerling, Jeff. "Ansible for DevOps." (2014).
8. Morris, Kief. “Infrastructure as code: managing servers in the cloud." O'Reilly Media, Inc., 2016.