Applied Cloud Computing
Uppsala University – Autumn 2020
Report for Lab 2

Li Ju

26th September 2020

I Task 1:

1 Part 1

Explain your findings in part 1. Are VMs slower than the physical machine? If yes, explain the reason. Are there alternative to VMs? How do you compare them with VMs? Keep the answer fairly short, limit it to a few sentences, max 1/4 of a page.

[Answer]
5 tests are done on the virtual machine: the average and the standard deviation of 5 running times are 21.632s and 0.133s, respectively. Comparing with 20.856s on physical machine, VMs are slightly slower than physical machines, because all instructions must go through an additional virtualization layer to reach CPU from their operating system, while instructions on physical machine will be passed to CPU directly by OS.
Container technique is an alternative to virtual machine, virtualizing an operating system instead of hardware. To compare their performances, both a container and a virtual machine can be assigned identical hardware resources, do identical testing and compare their running time.

2 Part 2

task 0

1. What version of the API are we using?
2. Explain how the communication works in OpenStack?
3. Can we use EC2 and S3 APIs to communicate with OpenStack?

[Answers]

1. We are using openstack 3.18.0 (Stein)
2. Openstack is a RESTful API, sending URL requests to the service to issue commands with HTTP[1]. Among different components, keystone is the service responsible for identification. With the authenticated keystone session, one can access compute service via nova, access block storage service via cinder, or other services.

3. No, because EC2/S3 are provided by AWS, whose APIs have different syntax with OpenStack. Therefore, they cannot communicate with each other.

task 2

1. Explain the output?
2. What is contextualization?
3. What language is use to prepare CloudInit configurations?
4. What are the variants of CloudInit package?
5. Can we run CloudInit scripts without booting an instance?
6. What limitation you can anticipate with the CloudInit package?

[Answers]

1. The output is a group of characters in cow shape, printed by cowsay. The python script wrapped cowsay on virtual machine as a service so that it can be accessed remotely.
2. Contextualization is the step to customize one's virtual machines automatically when booting using a configuration file.
3. In this case, CloudInit uses cloud config files, which is formatted in yaml language.
4. For virtual machines, Ansible is an alternative to CloudInit. For Docker containers, Docker Compose can be used to customize containers.
5. To run CloudInit scripts on new instances, new VMs must be booted. After new instances are booted, it is also possible to run CloudInit scripts again - "sudo cloud-init clean && cloud-init init".
6. The sizeo of script for CloudInit is limited to 163841 bytes.

task 4

1. What language is used with the Heat service to define the templates?
2. What are the advantages of using templates rather than the APIs?
3. Explain the different sections in the templates?

[Answers]

1. The templates used for heat service are defined in yaml language.
2. Like a recipe, template simplifies the process of initializing multiple virtual machines and other resources, and makes the procedure in a more readable, organized and transferable way, rather than deploying a stack step by step via command line.
3. (a) parameters: the section of parameters defined some parameters which may be required when allocating resources, including the flavor, base image, key pair name and public network
   (b) resources: the section of resources describe resources required when creating a stack, including network resources (router, private and public network, floating ip) and computing resources (instances)
   (c) output: the section of outputs describes information which need be returned when finishing allocating resources.

task 5

1. In what category of virtualization do containers fall?
2. What are the other frameworks that provide container technology. Write at least two name.
3. Explain the provided Dockerfile. What does it do? How does it work? Write a brief (one line) description about each line in the Dockerfile.
4. Write a brief (one line) description about each command used in Step-2-2.
5. What is dockerhub? Write a brief description of how can we use dockerhub for our newly build CSaaS container?
6. Write a CloudInit script that contextualize a VM based on the steps (Step-1 and 2) mentioned in this task.

[Answers]

1. Container is on the virtualization level of operating system.
2. Besides Docker, CoreOS rkt, Mesos, LXC, OpenVZ are also container tools.
3. Dockerfile is like a recipe telling Docker how to build a customized container image step by step. With the docker image, docker can run a container based on the image. Comments for dockerfile is shown as following:
   (a) FROM ubuntu: select the base image as ubuntu:latest
   (b) RUN apt-get update: update apt-get lists
   (c) RUN apt-get -y upgrade: upgrade apt-get and automatically choose yes for upgrading options
   (d) RUN apt-get install -y git: install git with apt-get and choose yes in default
   (e) RUN apt-get install -y python-pip: install pip with apt-get and choose yes in default
   (f) RUN pip install –upgrade pip: upgrade pip with pip
   (g) RUN pip install flask: install flask framework with pip
   (h) RUN apt-get install -y cowsay: install cowsay with apt-get and choose yes in default
   (i) RUN git clone https://github.com/TDB-UU/csaas.git: clone the repository for deploying cowsay as a remote service from https://github.com/TDB-UU/csaas.git using git
   (j) WORKDIR /csaas/cowsay: switch working directory to /csaas/cowsay
   (k) EXPOSE 5000: expose port 5000
   (l) ENV PATH="$PATH:/usr/games/": set the environment variable PATH as "/usr/games"
   (m) CMD ["python","app.py"]: run the app.py script (script for deploying cowsay as a remote service)
4. (a) docker build -t cowsay:latest .: build a docker image (docker build) from the dockerfile in the current directory (the last "."), and the image name is in {name:tag} format as cowsay:latest.

(b) docker run -it cowsay: run a docker container (docker run) from the image cowsay (we built just now), and attach the container an interactive (-i) psudo-tty (-t).

(c) docker run -d -p 5000:5000 cowsay: run a docker container from the cowsay image background (-d) and map the container system's port 5000 to host machine's port 5000 (-p 5000:5000).

5. Dockerhub is a library to share docker images with others: one can upload his/her images and download others' conveniently. When we are building the CSaas container, we ARE using dockerhub: the base image ubuntu is pulled from dockerhub. Also, we can push our dockerfile to dockerhub (e.g. liju666/cowsay:latest). For anyone who wants to build the service, (s)he only needs to run "sudo docker pull liju666/cowsay && sudo docker run -d -p 5000:5000 liju666/cowsay", the service can be deployed on his/her own machine.

6. The CloudInit code is attached with the report.

II Task 2

1 Part 1

Questions on the notebook file are finished and the PDF file of the notebook is attached.

2 Part 2

Cloud computing is an emerging computing paradigm from the last decade. It allows users to access a shared pool of a huge amount of resources on demand. Correspondingly, the number of cloud computing vendors increases rapidly as well. They offer clients numerous options when choosing services of IaaS, PaaS and SaaS level. For clients, it is highly possible that they want to immigrate their deployed services from a provider to another, or deploy services with resources across different vendors. Therefore, the concept of cloud interoperability is rising.[2]

Lack of interoperability has many causes:

1. Generally, different vendors have different SLA(service level agreements), which may cause some problems when operating their resouces together.
2. On IaaS level, different vendors often have different resource management system and different APIs for the system. If you use resources from different providers, orchestration is often hard to do.
3. Resource components from different providers may not be able work together because of different underlying implementations/mechanism.
4. On Paas and Saas level, based on their own infrastructures, many vendors have their own unique services. Once your applications are based on these unique services, it makes it harder to immigrate to other cloud providers.[3]

To improve the interoperability of cloud computing, there are many examples. Generally they can be categorized as three approaches: provider-centric, client-centric and resource management framework supporting interoperability.[3][4]

Provider-centric interoperability Provider-centric approach is actually standarization. There are many organizations are efforting to standarize cloud computing. These standards cover access mechanism, virtual appliance, storage, network, security, etc.[5][6][7] However, standarization of cloud computing requires cloud vendors to follow these standards, which may take a long time.

Client-centric interoperability Instead, client-centric interoperability is a more practical approach for clients to manage their resources of different vendors. Clients can create an intermediate layer above providers APIs. The intermediate layer abstracts resources and offer them to users. Users can demand these resources using the intermediate's API regardless of the actual provider of resources. Libcloud and δ-cloud are such APIs clients can use. However,

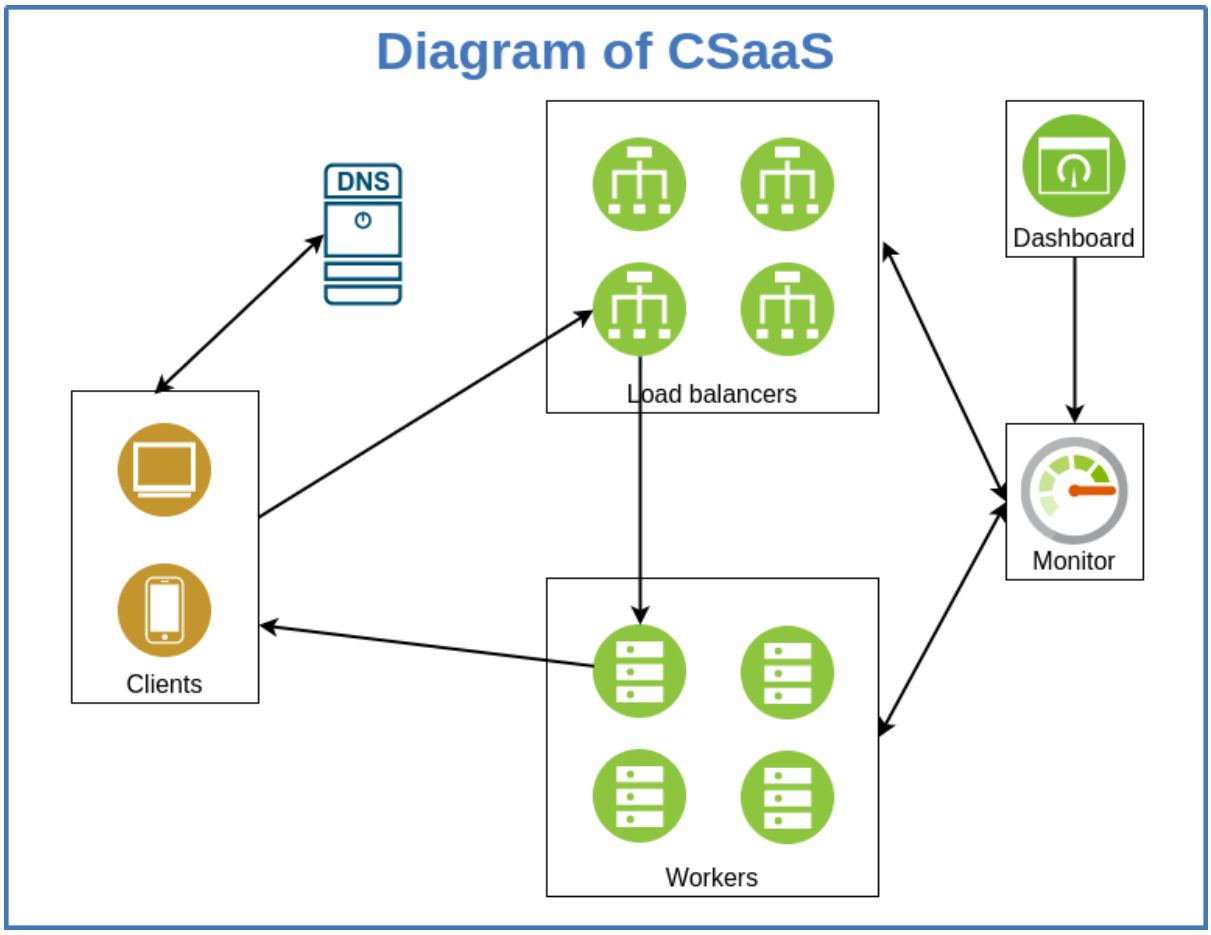
such intermediate layers cannot expose provider-specific services, but only common services from different vendors.[8][9]

Resource management framework supporting interoperability This approach for interoperability is actually encouraging cloud service providers to use those well-implemented open source resource management framework for cloud computing, like OpenStack, Eucalyptus and OpenNebula. It often cost much less to immigrate services from different vendors using the same underlying management framework.

All approaches above are mostly for IaaS level. For higher level like PaaS and Saas, it is often much more difficult to immigrate applications based on these services. To improve cloud interoperability of your own application, it is strongly recommended to use less platform-specific services.

"Cloud computing has become a lot like the Hotel California: Once you pick a provider you can check out anytime you want - but you can never leave."[10]

3 Part 3



The diagram of CSaaS (CowSay as a service) is shown. When a client makes a request, firstly the DNS server will resolve an IP address of a load balancer nearby. There are several load balancers distributed in different Zones (e.g, Europe, North America, East Asian, etc), and

HAproxy is deployed on each of them. Each load balancer is connected with a cluster of worker instances in its own region. After received the request from the client, the load balancer will forward the request to a low-loaded (comparatively) worker of the cluster it takes in charge and the worker will response to the client directly. This strategy can achieve fast response for clients from different regions and load balance within a certain region.

Moreover, a monitor instance is also designed to monitor and scale load balancers and workers. With collected workload data, some strategies could be made to dynamically adjust the number of workers (e.g. Follow the Sun). Also a dashboard is designed, and the company can manually scale workers on their own need on the dashboard. This strategy ensures scalability of the service (both automatically by strategy or manually).

The communication between each part is based on some lightweight protocols. And to deploy such a loosely-coupled application, contextualization and orchestration tools are required: Heat can be used to create a couple of organized virtual machines, and these VMs are customized by CloudInit scripts. On these VMs, containers will be used to run these micro-services. Kubernete will be used to deploy these Docker services between VMs.

References

[1] Jeff Cogswell. Introduction to the OpenStack API. https://www.linux.com/training-tutorials/introduction-openstack-api/, 2015.

[2] Zhizhong Zhang, Chuan Wu, and David WL Cheung. A survey on cloud interoperability: taxonomies, standards, and practice. *ACM SIGMETRICS Performance Evaluation Review*, 40(4):13–22, 2013.

[3] Tharam Dillon, Chen Wu, and Elizabeth Chang. Cloud computing: issues and challenges. In *2010 24th IEEE international conference on advanced information networking and applications*, pages 27–33. Ieee, 2010.

[4] AV Parameswaran and Asheesh Chaddha. Cloud interoperability and standardization. *SETlabs briefings*, 7(7):19–26, 2009.

[5] Wikipedia contributors. Open virtualization format — Wikipedia, the free encyclopedia, 2020. [Online; accessed 24-September-2020].

[6] Wikipedia contributors. Cloud data management interface — Wikipedia, the free encyclopedia, 2020. [Online; accessed 24-September-2020].

[7] Wikipedia contributors. Open cloud computing interface — Wikipedia, the free encyclopedia, 2020. [Online; accessed 24-September-2020].

[8] libcloud. http://libcloud.apache.org/.

[9] Deltacloud. http://deltacloud.apache.org/.

[10] Alexander Haislip. Breaking through cloud addiction. https://techcrunch.com/2012/12/01/netflixs-amazon-cloud-addiction/.