CS312 Solutions #6

March 13, 2015

Solutions

1. (1pt) Define in detail what a load balancer is and what problem it's trying to solve. Give at least two examples of where using a load balancer might be useful, and describe why.

   Load balancers provide a means to distribute a workload across multiple computing resources. Its typically used for web sites, but can be used for about anything IP based. It can also offer redundancy by providing health checks for backend servers. Examples can vary but may include situations such as scaling out a service to deal with more load or providing weighted balancing of resources based on computing capacity of the backend node.

2. (1pt) Describe pros and cons of each type of load balancer.

   (a) **Round Robin DNS:**

   - **Pros: Easy to implement.**
   - **Cons: doesn't handle automatic failover well. In addition DNS caching can become a problem since client's DNS servers may not update the DNS record in a timely fashion. Also some client resolver libraries may always return records in an alphabetical order instead of randomized.**

(b) **Software TCP/UDP load balancing:**

- **Pros: Generally easy to implement depending on which solution you use. It also offers a lot of flexibility in how you route your traffic and provide health checks.**
- **Cons: Requires dedicated nodes and setting up redundancy with those nodes.**

(c) **Hardware TCP/UDP load balancing:**

- **Pros: Generally offers more throughput and dedicated hardware to deal with SSL offloading. Also provides more official support if things go wrong.**
- **Cons: Typically closed source, expensive and sometimes difficult to use and administrate.**

3. (1pt) Describe what Round-Robin DNS is and when it might be useful to use it.

   Round-Robin DNS (RRDNS) is a technique to provide load distribution using DNS by assigning multiple IP addresses to the same A or CNAME record. When a client queries the record, the resolve receives multiple IP addresses and chooses one to connect to. Its typically useful in situations where you don't mind the delay of downtime if one of the services goes down. It's also useful if you just need simple load balancing without needing to track sessions. Another reason is balancing traffic across geography and you don't want to have everything go throught the same load balancer. In addition, if your application doesn't work well with standard load balancer, RRDNS can be a useful alternative.

4. (1pt) Say you're serving a web application that needs to keep track of its sessions for users. Now you have this application distributed across 100 machines. What scheduling algorithm would you choose with HAProxy for this application? What other information might you need to make a better decision?

The *Source Connection* algorithm would likely be the best choice however others might be a good choice too depending on the application needs. Additional information you may need will vary, but here are a few we think are important to know:

(a) **How does the application use session cookies?**

(b) **Are there any specific URL's in the application that are static only? That would allow us to offload or redistribute that load using CDN's or using something proxy cacher/acclerator such as Varnish.**

(c) **What does the traffic load profile look during a course of a day? Do we need to run all 100 machines all the time, or can we turn them off/on as we need them?**

(d) **Does the application provide a health check page we can use instead of just checking on the HTTP status code?**

5. (1pt) Describe what persistence is in the context of load balancers. Why is it important? What are some issues with using persistence?

   Persistence is described as keeping a connection with a specific client connecting to the same server and backend servers throughout the session. It's important because without maintaining a persistent connection, the user may log out switching to a different backend server. One problem is if the server the user is connected to goes down, then the user will have to re-login when it connects to another backend server.

6. (5pts) Create a new OpenStack VM. We will be setting HAProxy to serve a site that uses 8 different application servers that are simulated with a simple python one-liner. Here are the details about each application server:

   - We have two blog applications running on ports 8000 and 8001 that will show "Blog Page".

- We have one admin application running on port 8002 that will show "Admin Page".
- We have five www applications running on ports 8003, 8004, 8005, 8006 and 8007 that will show "WWW Page".

Complete the following tasks:

(a) Install and setup HAProxy. Setup the `global` and `defaults` sections like we did in class. For now don't add any frontends or backends. Also make sure you setup logging with `rsyslog`.

(b) Setup the HAProxy admin port so we can see stats.

(c) Download this script (`http://cs312.osuosl.org/_static/hw/haproxy.sh`) and run it. This will setup a few simple HTTP servers using python to simulate a cluster of applications. Make sure you see several log files in `/tmp/hw6/logs` so you can see the output of the HTTP servers.

(d) Create a frontend on port 80.

(e) Create three backends called `blog`, `admin` and `www` that connect to the ports mentioned above for each app (assume you're using `localhost` as the hostname).

(f) For the www backend, give the apps running on port 8006 and 8007 a weight of 100, while the others have a weight of 50.

(g) Setup acls for `/blog` that point to the blog app, and `/admin` that point to the admin app.

(h) Set the www backend as the default backend.

(i) Now try accessing the site. Do `/blog` and `/admin` show the correct content? Does the main page work properly? Access each URL several times to ensure the weighting is working properly.

Create a gzip tarball that contains the `haproxy.cfg` file and all of the log files in `/tmp/hw/logs`, the CSV file that is created on the HAProxy admin page and `/var/log/haproxy.log`.

The logs should show the weighting is working properly and accessing the backend urls correctly as well. The CSV file should mimic what you have configured below in haproxy.cfg.

```
global
    log         127.0.0.1 local2
    chroot      /var/lib/haproxy
    pidfile     /var/run/haproxy.pid
    maxconn     4000
    user        haproxy
    group       haproxy
    daemon
    stats socket /var/lib/haproxy/stats

defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option                  http-server-close
    option                  forwardfor except 127.0.0.0/8
    option                  redispatch
    retries                 3
    timeout check           2s
    timeout client          1m
    timeout connect         10s
    timeout http-keep-alive 10s
    timeout http-request    10s
    timeout queue           1m
    timeout server          1m
    maxconn                 3000

listen admin-stats
    bind 0.0.0.0:22002
    mode http
    stats uri /

frontend http
    bind 0.0.0.0:80
    acl url_blog path_beg /blog
    acl url_admin path_beg /admin
    use_backend blog if url_blog
    use_backend admin if url_admin
    default_backend www

backend blog
    reqrep ^([^\ :]*)\ /blog[/]?(.*) \1\ /\2
    server blog1 localhost:8000 check
    server blog2 localhost:8001 check
```

```
backend admin
    reqrep ^([^\ :]*)\ /admin[/]?(.*) \1\ /\2
    server admin1 localhost:8002 check

backend www
    server www1 localhost:8003 weight 50 check
    server www2 localhost:8004 weight 50 check
    server www3 localhost:8005 weight 50 check
    server www4 localhost:8006 weight 100 check
    server www5 localhost:8007 weight 100 check
```

7. (1pt) Define a hypervisor. If you had to classify KVM, would you classify it as a type 1 or type 2 hypervisor? Explain your reasoning in detail.

   A hypervisor is responsible for creating and running virtual machines. It is also responsible for their management and execution. Since KVM is directly integrated into the kernel, it is reasonable to call it a type 1, or bare-metal hypervisor. However, the linux kernel is also typically part of a more generic operating system than most bare-metal hypervisors, and as such can sometimes resemble a type 2, or hosted hypervisor.

8. (1pt) Can you use KVM to virtualize a machine running a different processor architecture than the host? Explain your reasoning in detail.

   No, because KVM is what is traditionally called hardware-assisted virtualization, which requires hardware support and is specific to that set of hardware. KVM does, however, run on multiple architectures, but one could not run a KVM hypervisor on x86 with a PowerPC virtual machine. In order to do that, emulation is required.

9. (1pt) Explain the relationship between KVM and QEMU in detail.

   KVM is a kernel module that converts the Linux Kernel into a virtual machine monitor, or a hypervisor. It interfaces with the hardware virtualization features on the host processor and creates a device called `/dev/kvm` which allows user space ap-

plications access to the hypervisor features. QEMU is a user-space application that is used to emulate an operating system virtually. QEMU can be configured to use KVM which enables hardware accelerated virtualization and a hypervisor. It will access the `/dev/kvm` device created by the kernel which provides an interface to the hypervisor.

10. (1pt) Name three primary differences between KVM and Xen.

    (a) **KVM requires hardware support while Xen does not**

    (b) **Xen has a more mature code base than KVM**

    (c) **Xen uses a micro-kernel hypervisor while KVM has it built into the Kernel**

11. (2pts) Do some more research online regarding the differences between KVM and Xen. Based on what you have researched, which hypervisor would you choose if you were creating a virtualization server? Explain your reasoning in a paragraph and provide links to the sources you found online. In addition, what would be a good reason to choose neither and go with something proprietary such as VMWare?

    This answer entirely depends on your research and how you reasoned your answer.

12. (1pt) How does Libvirt relate to KVM and OpenStack? What problem is it trying to solve?

    Libvirt provides an API interface to various hypervisors. Openstack interfaces with Libvirt which then interfaces with KVM. Libvirt is trying create a unified API for all hypervisors. Each hypervisor has their own API interface so Libvirt creates a unified one that then translates it to each hypervisors' API.

13. (3pts) For the following list of Openstack services, briefly explain what each does and why it is useful. Research two more not on this list, and give their codename, an explanation of what the service does, and an explanation of why the service is useful.

    - cinder

- glance
- keystone
- neutron
- nova
- swift

A **Cinder: Block Storage, it provides block-level storage space for the virtual machines.**

B **Glance: Image, it manages Images which Nova deploys as VMs.**

C **Keystone: Authentication and (some) Authorization. It provides a centralized Authentication service for all other APIs.**

D **Neutron: Networking, it manages networking between VMs and external to the cluster. It allows for Software-Defined Networking and other useful features.**

E **Nova: Compute, it manages virtual machines and the related hypervisors using libvirt. If you want to run virtual machines, you need it.**

F **Swift: Object/Blob store: Stores things that look relatively like files. Especially useful when backed by a CDN for serving static content.**

G **The other two answers will depend on which services you picked.**