Instaclustr Technical interview

1. Bash Scripting

First section

```
BaseDirectory=${1}

find "$BaseDirectory" -type f | while read filename;

do
    echo $filename
    NumberOfWords=$(wc -w < $filename) #Counting the words present in the file
    echo "Number of words within this file: $NumberOfWords"
    echo ""
done
```

Extension

Getting character frequencies taken from here Base directory set to make running on repl.it easier.

```
BaseDirectory=${1:-/home/runner/PossibleUsedFiletype}

find "$BaseDirectory" -type f | while read filename;

do
    echo $filename
    sed 's/\(.\)/\1\n/g' $filename | tr '[:upper:]' '[:lower:]' | grep '[a-z]' | sort | uniq -c
done
```

This will output a numeric histogram for each individual file. I would love to aggregate them, but not familiar enough with bash to do so. I would also love to make an ASCII bar chart, I saw a few implementations around the place, but again, this is beyond my bash skills.

2. General Administion

Connected to the server using PuTTY. First had to convert .pem into .ppk file [1]

Ran 'top -i' to get a view of current processes. All processes are idle, though somehow >85% of our memory is being used

```
top - 07:33:57 up 1 day,  8:36,  1 user,  load average: 0.00, 0.01, 0.05
Tasks: 103 total,   1 running, 102 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   1016324 total,   877840 used,   138484 free,    23044 buffers
KiB Swap:        0 total,        0 used,        0 free.    73240 cached Mem

PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
```

Ran 'ps -aux' to view all running processes by all users. This reveals that python is single-handedly using 70% of memory.

```

USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
....
root       202  0.0  0.0      0     0 ?        S<   Oct12   0:00 [ext4-rsv-conver]
root       225  0.0  0.0   4440   352 ?        Ss   Oct12   0:00 /bin/sh -e /proc/self/fd/9
root       226  0.0 70.8 747648 720412 ?       Ss   Oct12   0:04 /usr/bin/python /var/lib/.instaclustr/hun
root       604  0.0  0.2  10220  2876 ?        Ss   Oct12   0:00 dhclient -1 -v -pf /run/dhclient.eth0.pid
message+   816  0.0  0.0  39220   832 ?        Ss   Oct12   0:00 dbus-daemon --system --fork
....
```

Ran 'less /etc/passwd' to get a list of all users to check security. I don't see anything untoward here. There's a root user, and me as 'ubuntu', a whole lot of nologin users. I don't understand what the 'false' or 'sync' users are. Hopefully not indicative of something wrong?

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
libuuid:x:100:101::/var/lib/libuuid:
syslog:x:101:104::/home/syslog:/bin/false
messagebus:x:102:106::/var/run/dbus:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
pollinate:x:105:1::/var/cache/pollinate:/bin/false
ubuntu:x:1000:1000:Ubuntu:/home/ubuntu:/bin/bash
```

Used 'hostnamectl' to check version of linux. It's running ubuntu 14.04.6, which came out in March 2019 and reaches end of life in 2022. So slightly dated, but still in service. The Kernel dates back to 2015. I have no idea of kernels go out of date or are more tied to hardware?

```
   Static hostname: ip-172-31-15-229
         Icon name: computer-vm
           Chassis: vm
           Boot ID: d9bb9538f5ce4e18a36adfa376b354d2
  Operating System: Ubuntu 14.04.6 LTS
            Kernel: Linux 3.13.0-74-generic
      Architecture: x86_64
```

Ran 'dmidecode | more' to try to check BIOS status but was denied permission

Ran lscpu to see what we're running on. Looks like a single core, single thread intel chip. Unsure if this just means it's virtual (like I assume it is?). Running lshw shows that we're on Intel Xeon which definitely sounds server-y

```
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                1
On-line CPU(s) list:   0
Thread(s) per core:    1
Core(s) per socket:    1
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 63
Stepping:              2
CPU MHz:               2400.056
BogoMIPS:              4800.11
Hypervisor vendor:     Xen
Virtualization type:   full
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              30720K
```

Additional things I would like to do:

I would like to check the backup status of the machine, but I'd have to work out what backup utility it was using and don't see anything obvious. I would like to see what that python code is doing with all that memory, and possibly terminate it. You said there are 'a few' things wrong with this server, but I really only found that python program, so would love to know what I missed.

3. General Programming

Functional code and output can be found in the "fizzbuzz.ipynb" notebook. Code replicated here for ease of assessment. Code written in Python 3

```
 def fizzbuzz(*arg, length = 100):
    #A function to solve the classic Fizzbuzz problem in a flexible way for instaclustr technical interview
    #Inputs: unlimited tuples of divisor/replacement pairs, plus the length of the output (default 100)
    #Outputs: A heading detailing what our replacment keys are, followed by output to terminal of the solution to the problem.



    #Print the key pairs to console so that user can verify they're input in the correct order
    #TODO: Add error checking to ensure keypairs are input as tuples and that they are in the format [str, int]
    for pair in arg:
        print('replacing multiples of {} with {}'.format(pair[1], pair[0]))

    print('\n')

    #Main loop - python is single sided ranges, so need to add 1 to the length to get expected user result
    for i in range(1,length+1):

        #Initialize the output string to be blank
        output = ''

        #For each pair, if we are an exact multiple of the selected divisor, we add the given string to the output string
        for pair in arg:
            if i%pair[1] == 0:
                output += str(pair[0])

        #If the string is still blank after the previous step then we simply output the initial number
        if output == '':
            output = i

        #Print the output to console
        print(output)

fizzbuzz(['Insta', 3], ['clustr', 5])
```

4. Concurrency Programming

The main loop of this program generates threads that generate random numbers and call a synchronized function to put data into a buffer. The synchronized "addToBuffer" function takes an int as an input, adds it to a buffer, then performs the calculations to give the min/max/avg/mode through other functions. When I run the code, it seems like the buffer fills up and the calculations are done correctly, but sometimes the print output is drastically out of order, but showing a later variation of the buffer. I never could work out what was causing this.

```
 import java.util.Random;
import java.util.Arrays;
import java.util.Collections;



public class RunMethodExample implements Runnable{
    public static int[] buffer = new int[30];
    //Arrays.fill(buffer, 0);
    public static int next_write = 0;
   public void run(){
      while (true){
        Random rand = new Random();
        int randint = rand.nextInt(10)+1;
        addToBuffer(randint);

      }
   }
```

```
    public synchronized void addToBuffer(int value) {
         RunMethodExample.buffer[RunMethodExample.next_write] = value;
         int[] tempbuffer = RunMethodExample.buffer;
         int currentloc = RunMethodExample.next_write;
         RunMethodExample.next_write = (RunMethodExample.next_write + 1) % 30;
         int buffersum = Arrays.stream(RunMethodExample.buffer).sum();
         float bufferavg = getAvgValue(RunMethodExample.buffer);
         int buffermax = getMaxValue(RunMethodExample.buffer);
         int buffermin = getMinValue(RunMethodExample.buffer);
         int buffermode = getMode(RunMethodExample.buffer);
         System.out.println(Thread.currentThread().getName()+" added value " + value + " to loc " + currentloc + " in the buffer" +
         "\nmin: " + buffermin + " max: " + buffermax + " avg: "+bufferavg + " mode: " + buffermode +
         "\n" + Arrays.toString(tempbuffer) + "\n");
    }

    public static int getMaxValue(int[] numbers){
         int maxValue = numbers[0];
         for(int i=1;i<numbers.length;i++){
             if(numbers[i] > maxValue){
                 maxValue = numbers[i];
             }
         }
         return maxValue;
     }
     //Find minimum (lowest) value in array using loop
     public static int getMinValue(int[] numbers){
         int minValue = numbers[0];
         for(int i=1;i<numbers.length;i++){
             if(numbers[i] < minValue){
                 minValue = numbers[i];
             }
         }
         return minValue;
     }

     public static float getAvgValue(int[] numbers){
         float avgValue = numbers[0];
         for(int i=1;i<numbers.length;i++){
             avgValue += numbers[i];
         }
         avgValue = avgValue/numbers.length;
         return avgValue;
     }

     public static int getMode(final int[] n) {
         int maxKey = 0;
         int maxCounts = 0;

         int[] counts = new int[n.length];

         for (int i=0; i < n.length; i++) {
             counts[n[i]]++;
             if (maxCounts < counts[n[i]]) {
                 maxCounts = counts[n[i]];
                 maxKey = n[i];
             }
         }
         return maxKey;
     }

    public static void main(String args[]){

       int threadcount = Integer.parseInt(args[0]);
```

```
        System.out.println("executing with " + threadcount + " threads");

        for (int x=0; x<threadcount; x++)
        {
            Thread temp = new Thread(new RunMethodExample(), "thread " + x);
            temp.start();
        }
    }
}
```

5. SQL Programming

Part 1

```
SELECT movies.title, movies.release_year, movies.genre, movies.director from movies
WHERE movies.genre = 'action';
# under the current database schema we could simply select * from movies, but specifying the desired fields
# will continue to work if additional columns are added in the future
```

Part 2

```
SELECT DISTINCT actors.name, actors.birth_year
FROM actors
LEFT JOIN movie_cast
ON actors.name=movie_cast.actor
LEFT JOIN movies
ON movie_cast.movie=movies.title
WHERE movies.director = 'Wes Anderson';
#This actually ends up being all the actors
```

Part 3

```
SELECT movies.title, movies.release_year, movies.genre, movies.director
FROM movie_cast #
RIGHT JOIN movies
ON movie_cast.movie=movies.title
WHERE NOT movie_cast.actor = 'Bruce Willis' AND movie_cast.actor = 'Jeff Goldblum';
```

6. Using Cassandra

Spinning up a cluster as per the details in the email was fairly straightforward. I then started using the documentation here to connect, create a keyspace and upload data.

First I went to the connection info tab and found the IP addressess to connect to. I've already added my client IP to the firewall so I should be able to connect.

```
Public:
"52.44.0.178", "34.197.95.64", "18.205.179.31"
```

Being familiar with python, I've decided to use it to connect to the cluster. I first have to install the cassandra driver

```
conda install -c conda-forge cassandra-driver
```

From your extremely nice demo code in python I was able to directly copy the following and conncect to my cluster. This showed my 3 nodes and that I was connected properly.

```
 from cassandra.cluster import Cluster
from cassandra.policies import DCAwareRoundRobinPolicy
from cassandra.auth import PlainTextAuthProvider

cluster = Cluster(
    [
        "52.44.0.178", "34.197.95.64", "18.205.179.31" # AWS_VPC_US_EAST_1 (Amazon Web Services)
    ],
    load_balancing_policy=DCAwareRoundRobinPolicy(local_dc='AWS_VPC_US_EAST_1'), # your local data centre
    port=9042,
    auth_provider=PlainTextAuthProvider (
        username='iccassandra',
        password='3e40b7ec5ead196a4b4bfd9bf0285dff'
    )
)
session = cluster.connect()
print('Connected to cluster %s' % cluster.metadata.cluster_name)

for host in cluster.metadata.all_hosts():
    print('Datacenter: %s; Host: %s; Rack: %s' % (host.datacenter, host.address, host.rack))
```

Right, lots of assumed knowledge around what keyspaces are and why I would want one. I think this is giving me enough of a gist to create one. Looks like it's around the replication strategy for a given set of tables? Anyway, using the following code I now seem to be the proud owner of my own keyspace.

```
session.execute("CREATE KEYSPACE tutorialspoint \n"
                "WITH replication = {'class':'SimpleStrategy', 'replication_factor' : 3};")
```

Ok, now to upload the csv. I tried to execute the transaction created from uploading my csv to sql fiddle, but obviously there are enough differences between SQL and CQL that that didn't work. I've managed to now create the table by running the following code. I'll be the first to admit that with Hollywood's current release strategy, using the title as the primary key is not that safe, but it'll have to do for now.

```
session.execute("""

        CREATE TABLE tutorialspoint.instaclustr(

            title text PRIMARY KEY,
            release_year int,
            genre text,
            director text
            ) ;

                """)
```

We seem to have created a table, so now we need to add data to it:

```
session.execute("""
    INSERT INTO instaclustr
        (title, release_year, genre, director)
    VALUES
        ('Independence Day', 1996, 'action', 'Roland Emmerich');
                """)
```

Lets double check to see that some data is indeed in there:

```
 session.row_factory = dict_factory
rows = session.execute("""
        SELECT * FROM instaclustr;
                """)
rows[0]

out
{'title': 'Independence Day',
 'director': 'Roland Emmerich',
 'genre': 'action',
 'release_year': 1996}
```

Looks like data! Now we just need to format the rest correctly. I could probably find/write a nice function to take data from a dataframe or csv and execute these transactions for me, but I'm so close to the end now, so I'll just brute force it:

```
 session.execute("""
    INSERT INTO instaclustr
        (title, release_year, genre, director)
    VALUES
        ('Jurassic Park', 1993, 'action', 'Steven Spielberg');
""")
session.execute("""
    INSERT INTO instaclustr
        (title, release_year, genre, director)
    VALUES
        ('Die Hard', 1988, 'action', 'John McTiernan');
""")
session.execute("""
    INSERT INTO instaclustr
        (title, release_year, genre, director)
    VALUES
        ('The Player', 1992, 'comedy', 'Robert Altman');
""")
session.execute("""
    INSERT INTO instaclustr
        (title, release_year, genre, director)
    VALUES
        ('The Grand Budapest Hotel', 2014, 'comedy', 'Wes Anderson');
""")
session.execute("""
    INSERT INTO instaclustr
        (title, release_year, genre, director)
    VALUES
        ('Moonrise Kingdom', 2012, 'comedy', 'Wes Anderson');
""")
session.execute("""
    INSERT INTO instaclustr
        (title, release_year, genre, director)
    VALUES
        ('Ghostbusters', 1984, 'comedy', 'Ivan Reitman')
    ;
""")
```

And a final check to see that it's all there:

```
session.row_factory = dict_factory
rows = session.execute("""
        SELECT * FROM instaclustr;
                """)
for row in rows:
    print(row)

result:
for row in rows:

    print(row)



{'title': 'The Player', 'director': 'Robert Altman', 'genre': 'comedy', 'release_year': 1992}
{'title': 'Jurassic Park', 'director': 'Steven Spielberg', 'genre': 'action', 'release_year': 1993}
{'title': 'Moonrise Kingdom', 'director': 'Wes Anderson', 'genre': 'comedy', 'release_year': 2012}
{'title': 'Ghostbusters', 'director': 'Ivan Reitman', 'genre': 'comedy', 'release_year': 1984}
{'title': 'Independence Day', 'director': 'Roland Emmerich', 'genre': 'action', 'release_year': 1996}
{'title': 'Die Hard', 'director': 'John McTiernan', 'genre': 'action', 'release_year': 1988}
{'title': 'The Grand Budapest Hotel', 'director': 'Wes Anderson', 'genre': 'comedy', 'release_year': 2014}
```

All code can be executed through the jupyter notebook 'cassandra.ipynb' found in this git.

7. Cloud Computing Theory

A. What is the difference between container based virtualization and hypervisor based virtualization?

In a nutshell hypervisor virtualization will emulate a full OS, creating virtual hardware for it to run on. A container system will run on top of a single OS and run only the application, without the overhead of virtualising an OS. This makes containers much more efficient. A fully tuned container system should see 4 to 6 times as many server instances on the same hardware than a hypervisor VM [1]. The development of containers has lead to 'microservices'. Since each container contains only the binaries and libraries required, rather than an entire OS they can be extremely lightweight, allowing a traditional large app to be split into multiple microservices that all communicate with each other[2].

B. Explain the idea of Immutable Servers and discuss the pros and cons.

Immutable servers are, as they sound, servers that are never changed or updated. Instead of making a config change on a live server a new image is built and a new server stood up to enact the change. This allows for server testing before go-live[3], and ensures that there won't be errors during the upgrade process that mean our server is in a partial state, with some componants upgraded but some failed. Immutable servers do raise challenges in the deployment and testing process[4]. Since a new server image needs to be stood up in test, automated test run, then assuming the tests pass, it must be stood up again in production. This can be a time consuming process, particularly when the automated testing becomes heavy. It also requires that data be externalized, as in most instances it is no good to replace an immutable one with a new one that doesn't have any of the existing data.

C. In Amazon Web Services, explain the difference between a Region, Availability Zone and Instance in relation to fault domains.

1. AWS Regions are distributed around the world and will have different latency depending where the region is being accessed from. Different regions also have different pricing structures and different services available.
2. AWS Availability Zones (AZ) are the building block that makes up the region. Each AZ is an isolated data centre, which allows servers running in a single region to have backups in other AZs[5]
3. Instance is a specific node of for example a cassandra cluster. This is the individual machine (or more likely VM or container) that the app is running in.

Fault domains are a way of understanding the redundancy in a server architecture built on AWS. A fault domain is a single point of failure, for example a fault domain at the AZ would fail if the entire AZ goes out, but not if one of it's constituant instances fails. If an application is running only on a single instance, then if that instance fails, either independantly or because of an AZ or Region failure, then the app itself will fail. By building nodes out across different AZs or even over different regions, higher resilliance and reliability can be achieved through redundancy.[6]

8. Networking Theory

A. Given a /8 base network (e.g. 10.0.0.0), show how to subnet it so there are at least 64 networks available.

64 is 2^6 so our mask must be represented by six ones and two zeros 11111100 which is represented as 252 in decimal. Thus our subnet mask is 255.255.255.252. This allows for 4 hosts per subnet.[1]

B. Explain what NAT/PAT are.

NAT is Network Address Translation while PAT is Port Address Translation. NAT requires a single external IP for every internal IP, while a PAT can take two internal

IPs and translate their data to the external network through a single IP by putting the data for each on different ports [2]

C. Explain what a VPN is.

A VPN is a Virtual Private Network. It allows for traffic to be encrypted on the host computer, sent over the computers internet connection to a host, which can then route the traffic to a different node, potentially in a different location. This has a few uses. It allows for secure data transfer over an ordinarily untrustworthy network, such as at a cafe or hotel. It also allows a user to spoof their physical location, for example to watch USA netflix from Australia. Lately VPNs have been advertised heavily for consumers to use in their homes. The pitch seems to be that you can't trust your giant corporate ISP not to use your traffic inappropriately but you can trust a tiny fly-by-night startup much more. Why one should be more trustworthy than the other has always confused me.

9. Memory Management

The CMS (Concurrent Marksweep Garbage collector) will scan the heap and mark instances that can be removed then clear the marked instances. The G1 garbage collector can divide the heap into multip equal parts and then prioritise the segments which have lesser live data. G1 is designed to work with larger heap and server class machines. It typically offers lower max latency at the cost of lower overall throughput. I found various experiments around the internet. This one found that the max latency was slightly improved at a cost of a massive reduction in throughput, while this found that CMS utterly dominated G1GC, both in total throughput, and max pause as G1 was too slow and had to implement many "stop the world" pauses, each lasting around 20 seconds. It seems wise to run tests on your own hardware and database to determine the best garbage collector for your needs.

Extension

There are a number of things that could cause the OutOfMemoryError in java. It could be that the available heap is not large enough to accomodate all live objects required by the program. It could be that there is a memory leak, where the application unintentionally holds references to objects in the heap, preventing them from being garbage collected. It is possible to dump a histogram of the heap, to have a look at what types of objects are taking up the memory, particularly taking multiple histograms over time and looking at which is growing continuously.