Cloud Computing Lab Session 1 Report

Mahsa Hadian, Dmytro Humeniuk, Tchanjou Njomou Aquilas and Moses Openja

February 16, 2020

Abstract

Cloud computing lab session aims to give students the hands on experience on the key concepts in cloud computing, the model of the cloud service, architecture and mechanisms in cloud platform, virtualisation among others. In our first lab session of cloud computing, we were introduced to AWS cloud service including EC2 and its basic operations such as creating, launching and stopping the instances as well as installing workbenches.

In this lab we report the results of our work on benchmarking six different Ec2 instances which include m4.large, c4.xlarge, c5xlarge, t2.2xlarge and t2.xlarge. As instructed in the assignment, we perform a number of operations to measure different metrics of the instance including CPU performance, memory, disk and network speed. Finally, we propose an application design for the case of data science company.

1 Introduction

AWS is a cloud service platform that provides building blocks for creating and deploying any form of application in the cloud. However, before using the instance it is a good practice to verify whether its parameters agree to the service provider documentation. Moreover, quite often the values in the specifications are exaggerated or shown for certain optimal conditions. During our lab 1 session of cloud computing we benchmark six instances that AWS provides through its web platform [1] mainly for the education purposes.
In our study we evaluate such parameters as CPU performance (the time it takes to perform certain operations, CPU occupation during the operations), disk performance (sequential reading/writing speeds, IOPS for different block sizes), memory (RAM speed, heap, stack performance) and network speed (download and upload speed). The characteristics of the instances that we extracted from AWS specifications are presented in the table 1 below.

Table 1: EC2 instances performance characteristics summary .

Name	CPU, GHz	CPU number	RAM, Gb	Storage, Gb	EBS bandwidth (Mbps)	Network performance	Price, $ per hour
m4.large	2.3	2	8	7.7	450	Moderate	0.1
c4.xlarge	2.9	4	7.5	7.7	750	Moderate	0.199
c5.xlarge	3	4	8	7.7	4750	Up to 10 Gps	0.17
m5.xlarge	2.5	4	16	7.7	4750	Up to 10 Gps	0.192
t2.2xlarge	2.3	8	32	7.7		Moderate	0.3712
t2.xlarge	2.3	4	16	7.7		Moderate	0.1865

The rest of our report is organised as follows: in section 2 we give an overview of the problem statement we address, section 3 presents the approach that was used to address the problem, section 4 gives the full results of our experiments. We finally conclude the results in the section 6.

2 Problem statement

- Choosing the right benchmark is still a challenge. The task of performing benchmark remains complex due to the fact that the maintainer is not sure of which benchmark tool to choose from.
- System administration or the customers are always presented with multiple instances, the performance of which they may not be sure about. However, it's important to know the weak and strong points of each machine to be able to use it for an appropriate application,

In our work, we therefore address the above listed problems by providing the quantitative proof to recommend the right benchmark and the ec2 instance choice based on the disk, CPU, memory and network performance.

3 Approach

In this section, we explain the tests we performed, parameter and benchmarks we used. It should be noted that, in order to record consistent results and reduce any noise in our experiments, for every we perform at least five prior tests to warm the the ec2 instance. We also repeat each parameter measurement five times and report the average value along with standard deviation and/or coefficient of variance.

3.1 Disk performance

Disk performance relates to the data transfer speed when writing/reading data to/from the disk. It can be characterized by such parameters as IOPS, which indicates the number of read and write operations per second and throughput and measures the number of bits read or written each second. The faster the disk can write/read data, the better performance it has.

To calculate the AWS instances disk performance parameters we use such benchmarks as Bonnie++, iozone, dd and hdparm. First of all, it's important to mention that such parameters as IOPS and throughput depend on the chunk (block) size used to read from/write to the file. Also, there are two types of read and write operations: sequential and random. Both of them take place in the working system.

3.1.1 Tests with Bonnie++ benchmark (writing/reading speed)

So, we first used Bonnie++ benchmark, which allows to specify the block size (we used the most frequently used block size values of 512 bytes, 4 kbytes, 8 kbytes, 16 kbytes, 64 kbytes and 256 kbytes). It calculates the number of such operations as sequential block reads/writes and seeks. The latter is useful to characterize the speed of random read/write operations. Every time we need to access a new block on a disk drive the actuator arm has to position the head on the correct track - that's a seek operation. Evidently, when performing random i/o operations, 'seeks' are performed quite often. Thus, the speed of this operation is directly proportional to random i/o operations speed. It's even more practical to use this characteristic, than a number of random reads/writes as such factor as the "random generator performance" isn't affecting the results. To perform the Bonnie++ test we used the following operation:

```
$ Bonnie++ -r 1500 -s 3300:{$block_size} -b
```

In this operation we specify the maximum RAM size to use with -r to 1.5 G and the file size with -s to 3.3 G. According to the benchmark specification, the file size should be twice the RAM of the machine to remove the influence of cache. However, the m4.large machine had the lowest amount of free storage memory of 3.3 G (checked with df -h command), which corresponds to 1.5 G of RAM to use for the test. So, we used this amount of memory and RAM as a compromise for all the machines. The results for the sequential reads were surprising. The values were higher than several Gbytes per second. This led us to assumption the we didn't succeed to minimise the cache influence.

To this end, we performed additional tests with the hdparm benchmark, which allows to measure disk reading speed with and without using cache.

3.1.2 Tests with dd benchmark (writing speed)

As noted earlier, the key role of the system administrator working with a Linux based operating systems in the cloud computing environment is to monitor the performance of a hard drive in terms of the read and write speed. To be sure about our previous tests, we perform more tests in this case we used *dd command* as one of the many options. We used a shell script that tests different parameters to measure both the server throughput, which to the write and read performance. The variable parameters used are the size of the block used by dd *(bs)* and (4): the number of blocks *(count)* for dd to create. In our experiments varied the parameters *bs* and *count* as described bellow: we used the bs values ranging between 1 to 4 and count ranging between 100000 to 3500000 while keeping the rest of the attributes as constant. Using the above configurations, we tired to identify the maximum writing speed. We further discuss the results of our experiments in section 4.1.2 of our results section.

3.1.3 Tests with hdparm benchmark (reading speed)

The hdparm provides a command line interface to various kernel interfaces supported by the Linux SATA/PATA/SAS "libata" subsystem and the older IDE driver subsystem [3]. It can be used to benchmark disk in order to find the read speed in term of regular and cached speed. It mainly takes only one parameter as input, which is the disk name, and starts reading data from that disk regardless of the filesystem.

For the disk benchmark, we tested the EBS drive on each Amazon instance. As different amazon instances have different disk names, we first got the EBS disk name by running the lsbk tool which gives information about mounted disk and their names on linux machines. The results of our experiments are presented and discussed in section4.1.3.

3.1.4 Tests with iozone benchmark (IOPS)

Input/output operations per second (IOPS) is defined as the standard unit of measurement for the maximum read and write operations in the hard-drive or SSD disks [2].

One of the benchmarks that allows to measure IOPS is izone. This benchmark is especially useful, when you need to test a big number of configurations. By specifying the parameters it iterates through the values automatically and gives the results in the csv friendly format.
As in the test with Bonnie++ we used different block sizes to test the i/o operations speed. We also experimented with the file size: we used files of 512 M and 1 G for the measurements. The difference for IOPS values for the two different file sizes was not significant (less than 1 %), so we chose the file size of 1 G. Unfortunately, the system got stuck when we tried bigger file sizes, which might be due to benchmark realisation faults, so we use 1 G file size as a compromise for all the instances.
We performed the test with the following command:

```
iozone -R -az -b iozone.xls -g 1 G -i 0 -i 1 -i 2 -I -n 512m -p -q 1024 -O
```

Parameter -az specifies auto mode, -R - excel friendly format, -g - file size, -i 0 1 2 - tests to do (writing, reading, random writing/reading), -I -use direct I/O for all file operations (tells the file system that all operations are to bypass the buffer cache and go directly to disk), -n - minimal file size of 512 Mb, -q - maximum block size of 1024 kb, -O - to specify IOPS .

3.2 Memory Performance

Apart from RAM size, one of the important parameters to evaluate its performance is the speed of memory manipulations, which can be measured in Mbytes transferred per second. It is also important to verify the endurance of stack, heap and data segments of the memory. We use ramspeed and stress-ng benchmark for the evaluation.

3.2.1 Memory tests with stress-ng benchmark

Stress-ng tool allows the system administrator to impose high CPU and stress test on a Linux based operating system. In this case multiple features can be tested. For our case, we chose it as the option for the memory tests in order to observe its performance under stress. We increased the number of "stressors" for heap, stack and data segment until a system memory error was detected.

3.2.2 Memory tests with Ramspeed benchmark (RAM speed)

Ramspeed benchmark is one of the numerous benchmarks provided by Phoronix-test-suite. Ramspeed measures effective bandwidth of both cache and memory subsystems. We used INTmem (integer) test to evaluate the memory speed of our instances. It's different from float point test

in that it runs the operations through ALU, which is, in general, more actively used, rather than FPU. INTmem is a test, closely tied the real world of computing [5]. It consists of four different subtests (Copy, Scale, Add, Triad) to measure different aspects of memory performance. "Copy" just transfers data from one memory location to another, "Scale" modifies the data before writing by multiplying with a certain constant value and "Add" reads data from the first memory location, then reads from the second, adds them up and writes the result to the third place. "Triad" is a merge of "Copy","Add","Scale".
We used "Triad" test as it provides a more universal metric.

3.3 CPU Performance

CPU performance can be evaluated by the number of calculations that were performed in a specific time or the by the time it took to perform a certain calculation. Also, the processor ability to parallelize the calculations should be taken into account.

3.3.1 CPU tests with Unixbench benchmark

The Unixbench performs a number of test (various concurrent integer and floating point calculations) to evaluate the CPU performance and gives the score for performing the tests with 1 CPU and the maximum number of CPUs the machine can provide. So, for each instance we present two scores to evaluate the CPU performance - one for performing a task with a single processor and one for doing the test with all the processors involved.

3.3.2 CPU tests with Sysbench benchmark

Sysbench is one of the most popular benchmarks, which can provide a quick characterization of system performance such as CPU, memory, file I/O, mutex performance, or MySQL. For the case of our experiments, we used sysbench to test the performance of the CPU on all the six ec2 instances.

To experiment and test for the CPU performance, we followed the same set of steps where, first we install the sysbench suite to all the six instances, warm up the workbench and the instance. We configured the workbench to use 2 simultaneous number of threads while varying the the maximum prime number to verify. In that case sysbench can verify prime numbers by doing standard division of the number by all numbers between 2 and the square root of the number. To have consistent results while avoiding noise, we chose to start with a prime value of 10,000 to 10,000,000, which worked for all the ec2 instances. We could not go beyond 50,000,000 otherwise one of the instance would crash and hence affect our results. To illustrate, we ran the tests of up to 5 times while using different prime-number attributes *cpu-max-prime* = {10,000, 100,000, 1,000,000, 10,000,000 }. The results of this test are presented in the section 4.2.

3.4 Network performance

Network performance is usually simply characterised by the network throughput as download and upload speed in Mbit/s. We used the SpeedTest benchmark, which runs a test against the closest speedtest.net server, to measure the specified parameters. The test was run by simply performing the command "speedtest".

4 Results

4.1 Disk performance results

In this section, we present the results of our tests on disk performances using "Bonnie++", "hdparm", "dd" command and "iozone" benchmarks.

4.1.1 Bonnie ++ (writing/ reading speed) results

Figure 1 shows the writing speed using broad ranges of six different *"block size"* of {512, 4k, 8k, 16k, 64k, 256k} and the corresponding coefficients of variation.

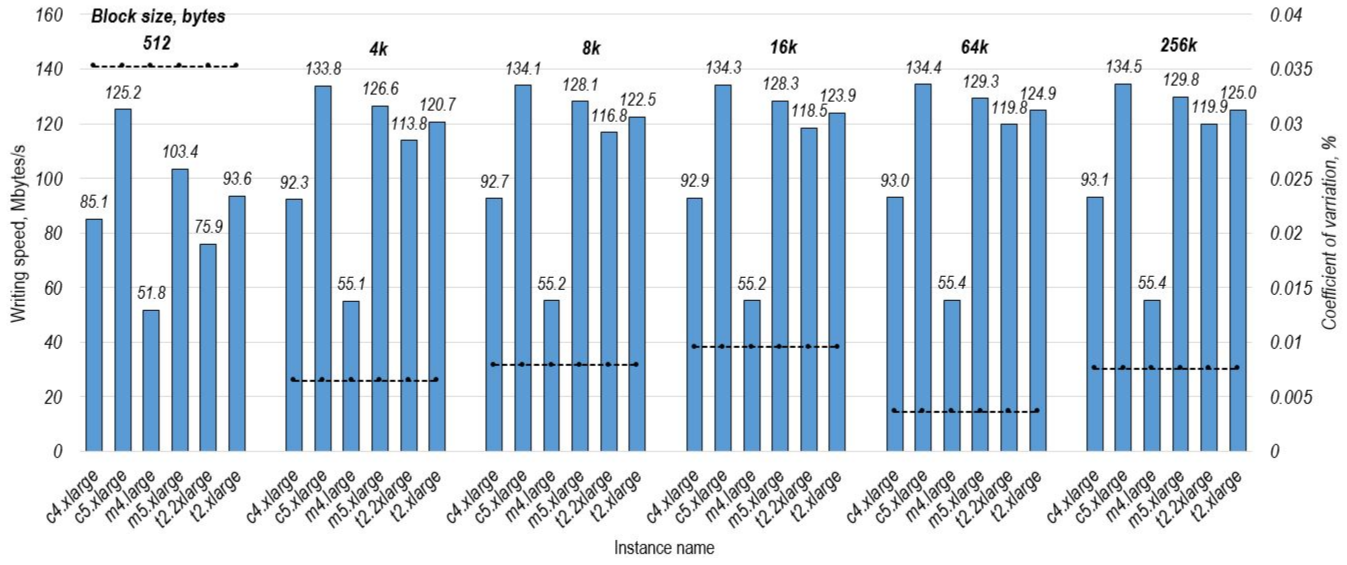


Figure 1: Writing speed results for different block sizes

We can see that writing speed is dependant on the block size and increases up to 7 % comparing minimal (512 bytes) and maximum (256k bytes) block sizes for each instance. It reaches its maximum value at the 4k block size. According to the results in the figure 1 instances m5.xlarge and c5.xlarge have the highest writing speed for all the block sizes. M4.large instance has the lowest writing speed. We conclude that the instances c5.xlarge and m5.xlarge are the most suitable for extensive writing i/o applications.

Figure 2 shows the disk read speed using the same block sizes as for the writing speed test presented above. Here we can see a sharp (up to 4x) increase of the reading speed for 4k block size comparing to 512 byte block. Our assumption is that at this point the caching was activated, which provoked such a change in the speed. We can see the further increase in the reading speed with the increasing block size, up to 200 % comparing 4k and 256k block sizes. At 64k block size the maximum speed is reached. Instance c5.xlarge demonstrates the highest reading speed

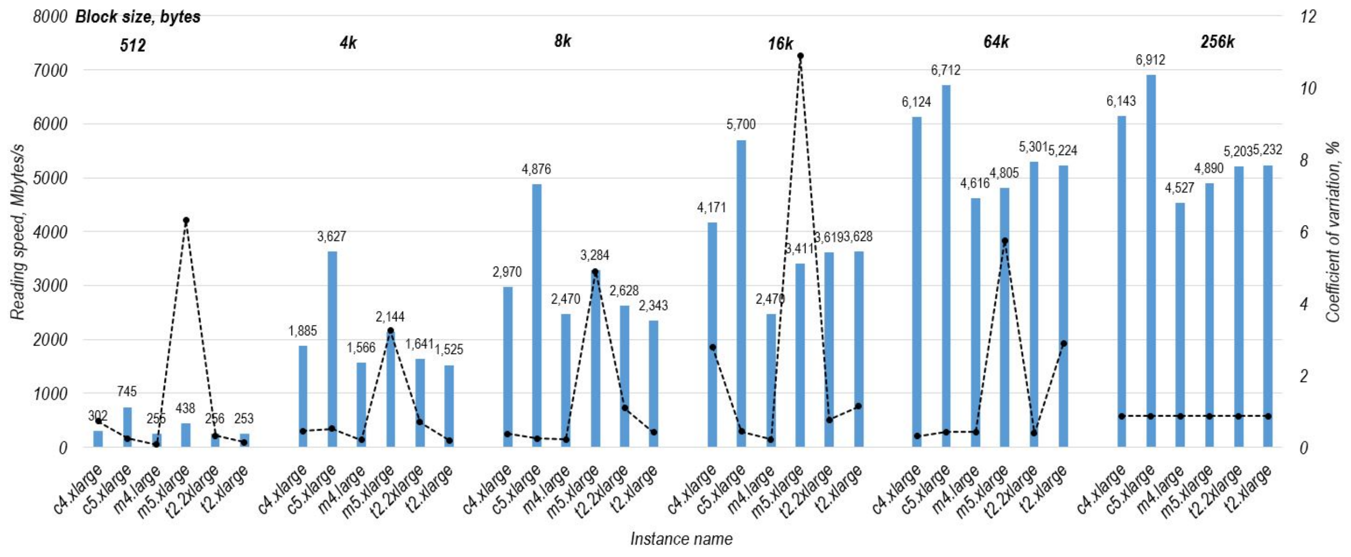


Figure 2: Reading speed results for different block sizes

of 6912 Mbytes/s and m4.large the lowest of 4527 Mbytes/s. M5.xlarge instance has second biggest reading speed for the block sizes of up to 16 kbytes. We see that for 64k and 256k blocks c5.xlarge and c4.xlarge have the highest speeds. However, according to AWS [1], the most used block sizes for operations don't exceed 16 kbytes. So, we recommend c5.xlarge and m5.xlarge for performing reading operations on smaller block sizes and, for special cases, c5.xlarge and c4.xlarge for bigger block sizes.

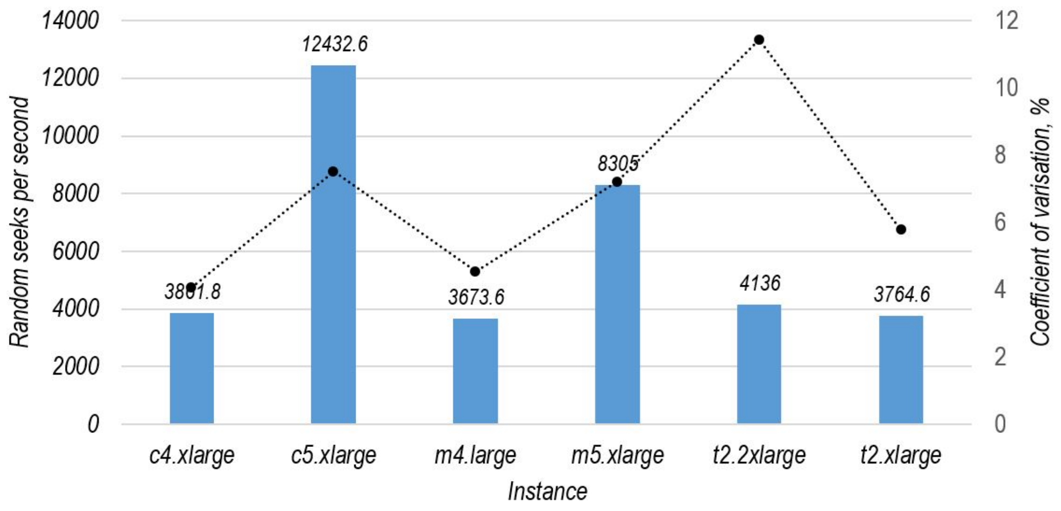
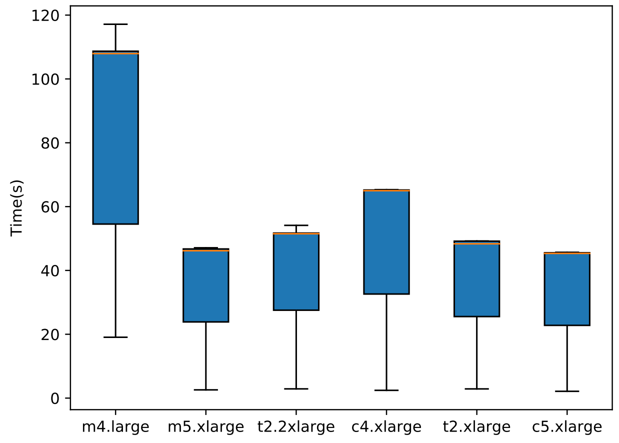


Figure 3: Random seek operations for different instances

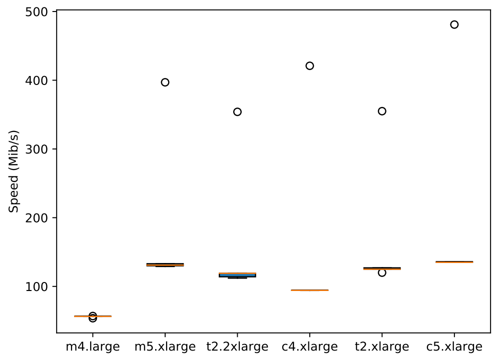
Figure 3 shows the results for the number of random seeks during disk i/o operations for 64 kbytes block size. We chose this block size, as it corresponds to the highest throughput value. The numbers illustrate the ability of the instances to perform random i/o operations, which is a frequent operation in a database. Instances c5.large and m5.large show the best random i/o performance with the number of 12432 and 8305 seeks per second respectively. Other instances have a noticeably lower performance, with the lowest value for m4.large instance of 3672 seeks.

4.1.2 dd command results

Figure 4 shows the IO write performance time in seconds and speed in Mib/s for the six ec2 instances.



(a) Instance IO Time



(b) Instance IO Speed

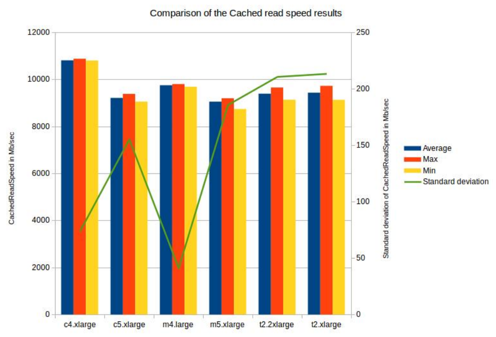
Figure 4: IO Performance measures

As shown in Figure 4, there is a non uniform performance in the IO read and write speed and time across the six ec2 instances, with m5.xlarge taking the minimal time (a) and the highest speed in (b). Comparing this results with Bonnie++, we still confirm the same disk IO performances in terms of writing operations.

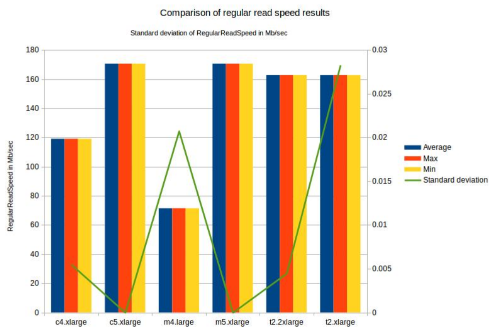
4.1.3 hdparm results

In this section we compare the disk performance in terms of cached and regular read speed across the six ec2 instances using the hdparm test, from the experiments in section 3.1.3. The hdparm test results are shown in Figure 5.

According to the results in Figure 5, c5.xlarge and m5.xlarge achieve the best results in term of regular read speed with the values of 174 Mbytes/s and 171 Mbytes/s respectively, while c4.xlarge and m4.large have the best cache read speed of 10 Gbytes/s and 9.5 Gbytes/s respectively. However, some instances are more stable than others. m4.large has the most stable disk cached read speed (The standard variation of its disk benchmark results is the lowest), while t2.xlarge has the most unstable EBS disk when it comes to the cached reads. For regular reads, c5.xlarge and m5.xlarge are the most stable, with a constant read speed for all the benchmarks. Also, we see that these two instances also have the best regular read speeds. The results fully agree with the Bonnie++ benchmark.



(a) Cached Read Speed comparison



(b) Regular Read Speed Comparison

Figure 5: Disk Read Speed comparison

4.1.4 Iozone (IOPS) results

We performed the iozone benchmark tests for the same block sizes as with Bonnie++. For the sake of simplicity we present the results for the 4 kbytes block size, which gives the highest IOPS. For the larger block sizes number of IOPS was decreasing.

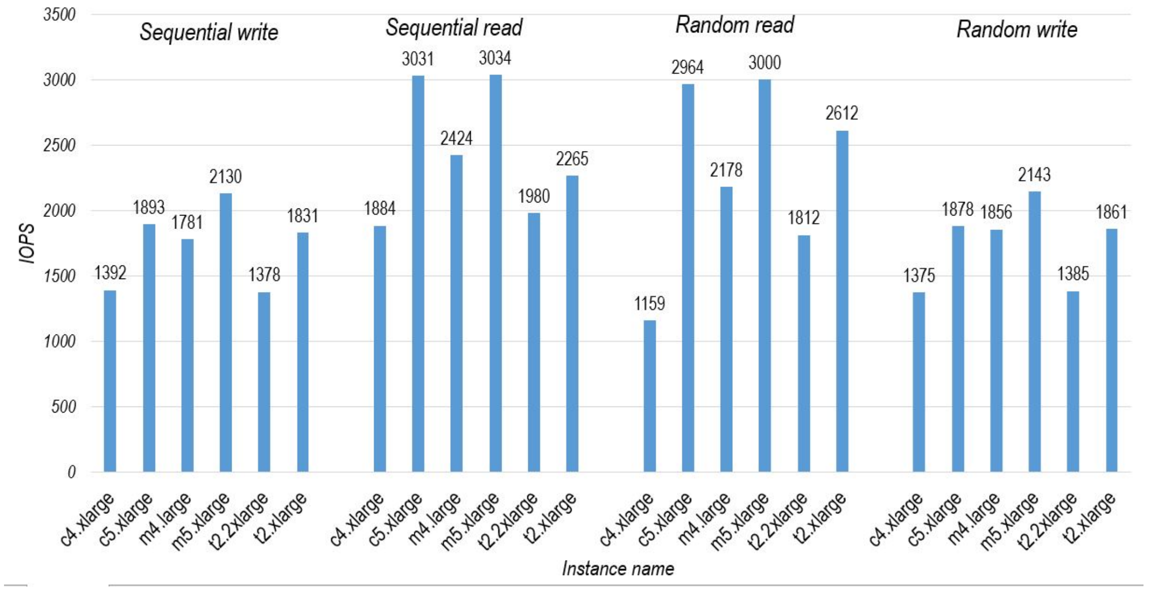


Figure 6: IOPS for block size 4k for different tests

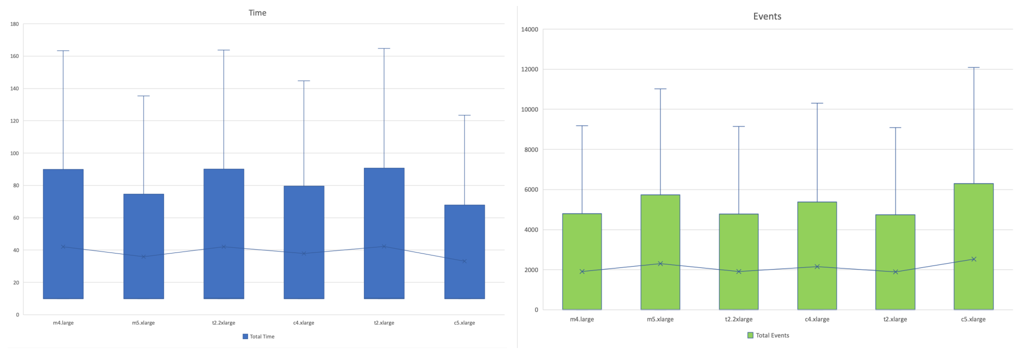
Figure 6 shows the IOPS results for *"Sential write"*, *"Sential write"*, *"Random read"*, *"Random write"*. The ec2 instances c5.xlarge and m5.xlarge demonstrate the highest sequential and random read and write performance. On the other hand, the instances c4.xlarge and t2.2xlarge have the lowest sequential read/ write and random read/write number of IOPS. This correlates with our previous results.

Comparing our four benchmark results, we can conclude with the following observation:

Observation 1: *ec2 instances m5.xlarge and c5.xlarge* show the best disk performance. We therefore recommend m5.xlarge and c5.xlarge for extensive disk I/O operations.

4.2 CPU performance results

Figure 7 shows the performance of the CPU as a measure of total time in seconds and the total events the CPU executed using Sysbench benchmark. We used broad range of our test cases



(a) Time (b) Events

Figure 7: CPU Performance

covering prime-limit = {10,000, 100,000, 1,000,000, 10,000,000}. As shown in Figure 7 (a), there is a difference in CPU performance time across the six ec2 instances, with m5.xlarge, c4.xlarge and c5.xlarge taking less time compared to the rest. This implies that their CPU performance is better comparing to the other three ec2 instances. Figure 7 (b) also tells us that the number of total events the CPU can execute within the given time period is the highest for ec2 instances m5.xlarge, c4.xlarge and c5.xlarge.

Figure 8 shows the CPU performance Unibench scores for both single and a maximum number of VCPUs for each of the ec2 instances. For single core performance c5.xlarge and m5.xlarge have the best scores of 1148 and 1073. When using multiple cores, t2.2xlarge gets the highest score of 4480. It isn't surprising as this machine has the biggest number of cores - 8. T2.xlarge has the second largest score of 2655.1. We recommend the three ec2 instances m5.xlarge, c5.xlarge and c4.xlarge as the best choice for non parallelizable tasks and t2.2xlarge and t2.xlarge for the operations that can be easily parallelized, like for example image or video stream processing.

Observation 2: *I*t is better to use m5.xlarge, c5.xlarge and c4.xlargefor single core applications. *t2.2xlarge and t2.xlarge* for multiple core optimized tasks.

4.3 Memory performance results

In this section, we present and discuss the results of the memory performance corresponding to the section 3.2 of our approach.

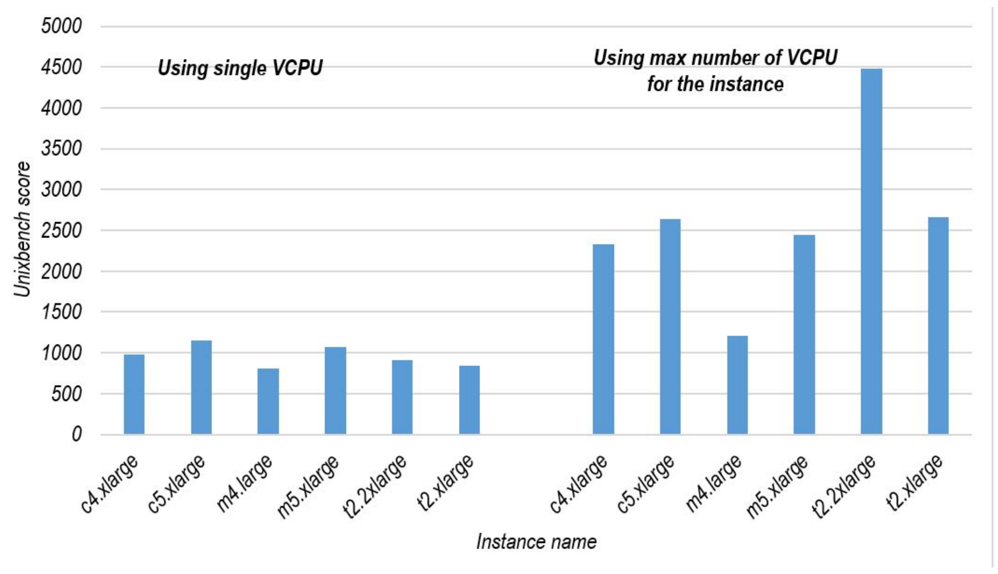


Figure 8: CPU performance using Unixbench

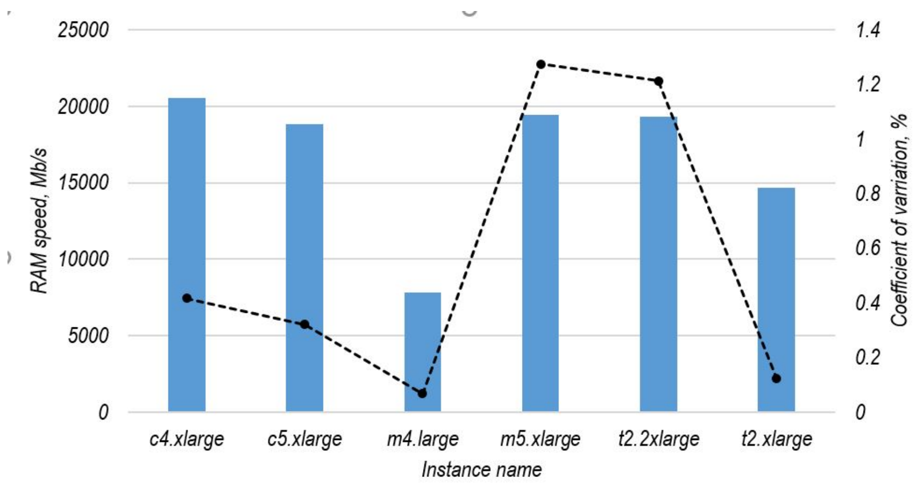


Figure 9: Ramspeed memory test results

Figure 9 shows the instances' RAM speed while executing extensive integer calculations. C4.xlarge, clearly, shows the best performance with the RAM speed of 20542 Mb/s. M5.xlarge and t2.2xlarge show the same level of performance at around 19400 Mb/s. M4.large has the lowest score - 7837 Mb/s.

Figure 10 shows the memory stress test results using three metrics: number of the heap, data segment (breakpoint) and stack stressors. We can see that the t2.2xlarger required the biggest amount of stressors to cause a memory error, which was over 1 million stressors. m4.large machine showed the weakest memory endurance needing 185000 stressors to crash.

Observation 3: *c4.large, m5.xlarge and t2.2xlarge* show the best performance memory performance considering the RAM speed and heap, stack, data segment endurance.

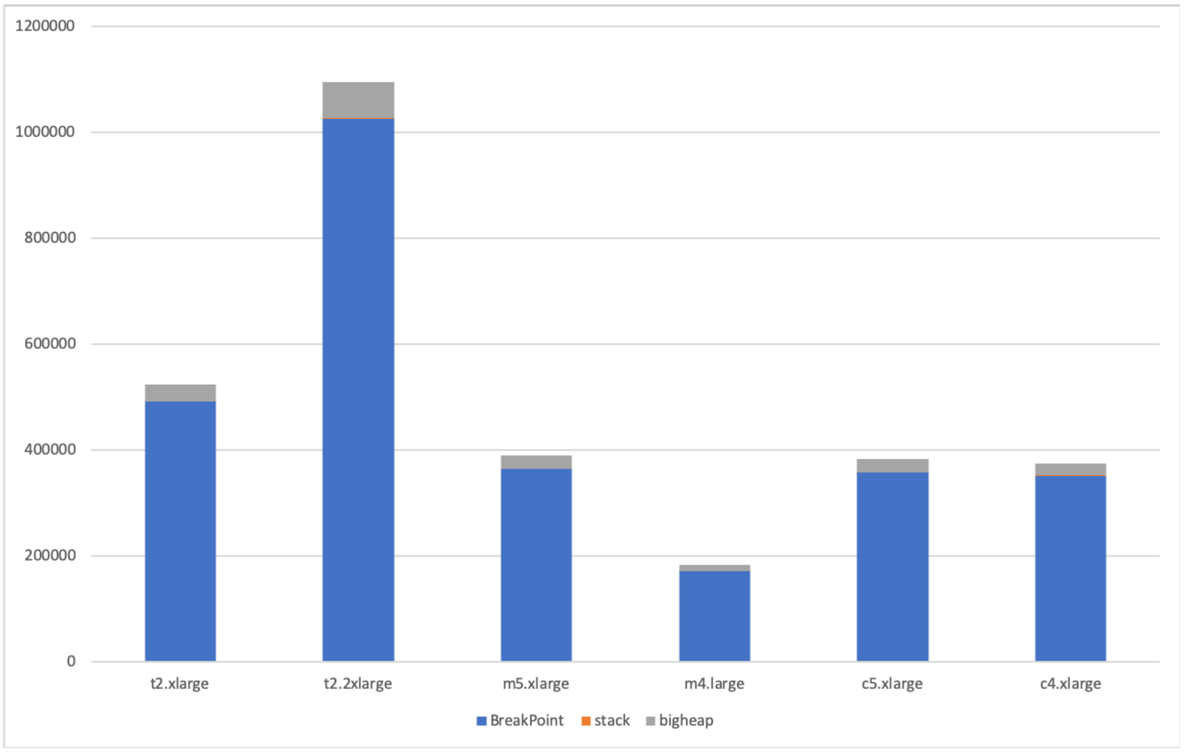


Figure 10: Memory Stress number

4.4 Network performance results

Here we discus the results of evaluating network performance in terms of the download and upload speed measured in Mib/s as explained in *3.4* of our approach section.

Figure 11 shows the network throughput for the different ec2 instances sorted in the order of the download speed. The coefficient of variation for measuring download speed didn't exceed 1 % and for upload speed - 0.01 % for all the instances. As shown in the Figure 11 the instance *c5.xlarge* has the highest download speed of up to 2857.3525 Mib/s and m4.large instance has the lowest download speed of only 500 Mib/s. We also observed that there is a minor difference in the network download speeds for the ec2 instances *"t2.2xlarge", "t2.xlarge" and "c4.xlarge"*, which is around 1000 Mib/s. The network upload speed is approximately the same for all the six instances and constitutes around 4.17 Mb/s.

We therefore recommend *c5.xlarge* as the best option, when high download speeds are required. For example, for making a chat application, where lag between the user actions and response is a priority.

Observation 4: *c5.xlarge* has the highest network throughput with the download speed of upto 2857.3525 Mib/s and m4.large instance has the lowest download speed of only 579.43 Mib/s

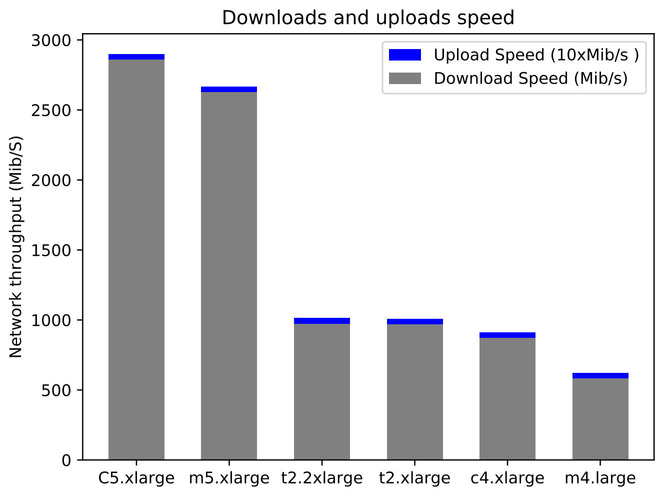


Figure 11: Comparison of Network throughput for each Ec2 Instance

5 Selecting the instances for the application

In this section, we provide our solution for selecting the appropriate instances for the specific task. Our finalized solution is shown in the fig.12

5.1 Storage Module

For storage, the application needs a storage optimised instance. In this plain all the instances have an EBS optimized storage of 8 Gbytes. Thus, we suggest to choose the least expensive ones to perform the storing function. According to the table 1, the m4.large and t2.xlarge instances have the lowest hourly prices of 0.1$ and 0.18$ respectively.

5.2 Compute Module

The compute module should be optimised to perform extensive disk I/O operations. For this task we have clear favourites: m5.xlarge and c5.xlarge. For working with large blocks of data we suggest also the c4.xlarge instance.

5.3 Visualisation module

Visualization module will be used to perform heavy computations and a number of parallelizable tasks. For this, we suppose,the best CPU and memory performance is required. t2.2xlarge instance showed the best multi core performance. Instances c4.xlarge and c5.xlarge have the highest scores for single core calculations and RAM speed. So, we recommend t2.2xlarge, c4.xlarge and c5.xlarge instances for this module.

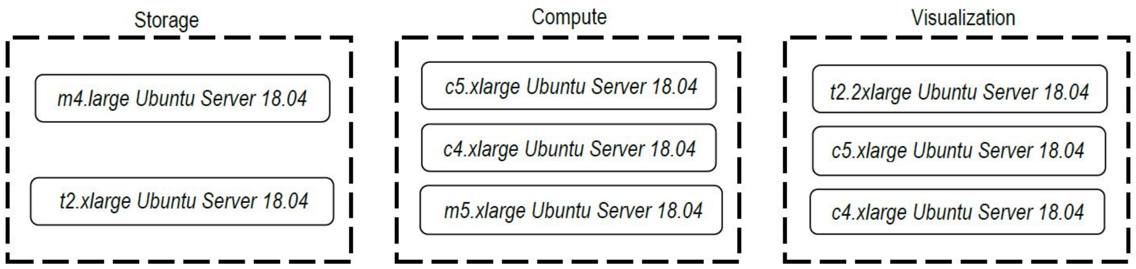

Storage
m4.large Ubuntu Server 18.04
t2.xlarge Ubuntu Server 18.04
Compute
c5.xlarge Ubuntu Server 18.04
c4.xlarge Ubuntu Server 18.04
m5.xlarge Ubuntu Server 18.04
Visualization
t2.2xlarge Ubuntu Server 18.04
c5.xlarge Ubuntu Server 18.04
c4.xlarge Ubuntu Server 18.04

Figure 12: Design architechture

6 Conclusions

In this report, we present the benchmarking results to compare the disk, CPU, memory and network performance for the six AWS EC2 instances.

Disk performance. To evaluate the highest disk performance we run extensive write/read i/o tests varying such parameter as the file block size. Varying the file size itself didn't produce any significant effect on the i/o operations speed. For the tested machines the maximum reading speed was reached at 64 kbytes block size and constituted around 6712 Mbytes/s for c5.xlarge (maximum) and 4618 Mbytes/s for m4.large (minimum). The maximum writing speed was reached at 4 kbytes and was the highest for c5.xlarge (133.8 Mbytes/s) and the lowest for m4.large (55.1 Mbytes/s). Overall, m5.xlarge and c5.xlarge showed the best disk performance.

CPU. We find the maximum value for CPU performance by running CPU exhaustive tests. One of them was verifying whether the number was prime or not. We found the maximum number of CPU operations per unit of time by increasing the prime number value. The compromise value was 10000000, when the 'number of events' for all the benchmarks was 1, which meant they were not able to handle the task. We then compared the time to finish the task. The c5.xlarge had the lowest execution time of 12.472 s and m4.large had the longest of 16.503. To evaluate the single core as well as multicore performance we used Unixbnech score as a metric. C5.xlarge had the highest value for single core performance of 1148, m4.large - the lowest of 812. For multi core, t2.2 xlarge showed the highest value of 4480, m4.large - the lowest of 1208.3.

Memory. We find the maximum RAM speed by performing calculations, which use RAM extensively. We used Ramspeed benchmark 'Triad' test for this. C4.xlarge has the highest RAM speed of 20000 Mb/s, while m4.large shows the lowest speed of 7837 Mb/s. We also test the ability of system memory of dealing with load on stack, heap and data segment. t2.2xlarge instance showed the best performance needing over a million "stressors" to cause an error, while m4.large needed the lowest amount of "stressors" - less than 200000.

Network. The network performance was evaluated by the download and upload speed of the machines. C5.xlarge machine showed the highest download speed of 2857.3 Mib/s and m4.large - the lowest of 579.43 Mib/s. The upload speed isn't significantly different for the instances and constitutes in average 4.17 Mb/s.

Problem solution. For storage applications we recommend using m4.large and t2.xlarge instances as the cheapest ones. For computation/data extraction applications we suggest c5.xlarge, c4.xlarge and m5.xlarge as the instances with the best disk i/o performance. For visualization services we would use t2.2xlarge, c5.xlarge and c4.xlarge instances as the ones with the best multi and single core performance as well as system memory speed.

References

[1] *AWSEducate*, Teach Tomorrow's Cloud Workforce Today https://aws.amazon.com/education/awseducate/.

[2] *Margaret Rouse*, IOPS (input/output operations per second). December 2016, https://searchstorage.techtarget.com/definition/IOPS-input-output-operations-per-second.

[3] *Michael Kerrisk*,HDPARM, March 2018, http://man7.org/linux/man-pages/man8/hdparm.8.html.

[4] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-volume-types.html.

[5] https://github.com/cruvolo/ramspeed-smp.