Show Submission Credentials

P3. Storage I/O benchmarking Storage I/O Benchmarking and Performance Analysis

41 days 23 hours left

Introduction

Tutorial

Tasks

Success

Introduction

Storage Vertical Scaling - Introduction

A virtual environment consists of various components based on interdependent resources such as processors, main memory, persistent storage, and network. Throughput and latency are significantly improved when accessing main memory compared to disk storage. If VMs are running I/O intensive applications, performance will be bound by the performance of the physical storage devices. When a VM runs out of physical memory, paging occurs and leads to increased load on the storage resources. As a result, the storage I/O bottleneck can affect the overall system performance. When it comes to performance tuning, you should consider optimizing the front-end server or back-end database as well as monitoring and tuning storage performance. Another aspect to consider is the performance of storage systems when an application is issuing sequential versus random I/O. Sequential disk accesses benefit from spatial locality, hence, the disk seek time is reduced.

In a virtual environment, any single read or write operation must be mapped from the guest OS to a physical device. Latency occurs during this process. Note that I/O intensive applications are sensitive to storage latency. Please keep this in mind during the twitter analytics web service project.

In the following sections, you will get familiar with common disk operations in Linux and use those commands to perform vertical scaling of storage devices by attaching different storage devices offered in AWS to instances and measuring the performance using some common benchmarking tools. After this, you will understand why physical storage devices where you store data also deserve your attention when you are trying to improve performance.

Sample Report and Analysis

If you don't have enough time to finish all the benchmarking iterations in this module, we offer you a sample report of the performance comparison to give you insight into the performance of storage I/O.

Scenario	Instance Type	Storage Type	RPS Range	RPS Increase Across 3 Iterations
1	t3.micro	EBS Magnetic Storage	171.12, 172.33, 189.34	Trivial (< 5%)
2	t3.micro	EBS General Purpose SSD	1649.65, 1709.24, 1729.24	Trivial (< 5%)
3	m4.large	EBS Magnetic Storage	527.70, 973.63, 1246.67	Significant (can reach ~140% increase with an absolute value of 450-700)
4	m4.large	EBS General Purpose SSD	2046.66, 2612.00, 2649.66	Noticeable (can reach ~30% increase with an absolute value of 500-600)

The RPS improves significantly when the benchmark is run on larger VMs and when using EBS with SSD. As you can also notice, the RPS increase across 3 iterations for m4.large is more significant than that for t3.micro. **The reason is an instance with more memory can cache more of the previous requests for repeated tests.** Caching is also a vital performance tuning mechanism when building high-performance applications.

Tutorial

Storage Vertical Scaling - Tutorial

Basic Disk Operations in Linux

On this page, there are several mini-howtos for accomplishing certain disk-related tasks in Linux. The instructions for the actual primer tasks that you need to complete will be presented later.

Since most of these commands require root access, you should prepend sudo to each command needed in this primer.

Here is the list:

```
/sbin/parted
/sbin/mkfs.ext4
/bin/mkdir
/bin/mount
/bin/umount
/usr/bin/sysbench
```

Working with EBS Volumes

The following video will demonstrate the use of EBS volumes with EC2 instances:

Video: Working with EBS Volumes

Warning

PATH system variable

You may notice that in the video the disk operation commands are executed by base names instead of the full path names, e.g. `parted` was used instead of `/sbin/parted` .

The `PATH` is the system variable that the OS uses to locate needed executables from the command line. When you run an executable without a full path, the OS will search it in the directories listed in `PATH` .

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/us
r/local/games
# which includes /sbin /bin and /usr/bin
```

In this primer, however, you **SHOULD** use **the full path names** when executing the commands due to our pre-defined `sudo` whitelist.

Tasks

Storage Vertical Scaling - Skill Building Tasks

File I/O Performance Benchmarks

In this part of the primer, you will be experimenting with "Vertical Scaling" for storage and instance types, by attaching Magnetic (i.e. spinning disks) and Solid State Drive (SSD) storage devices offered in AWS to two types of instances and measuring their performance for File I/O.

As a result, we expect you to gain insight into the benefits of employing faster storage systems such as SSD as well as using larger instances.

We want you to test the following scenarios:

Scenario	Instance Type	Storage Type
1	t3.micro	EBS Magnetic Storage
2	t3.micro	EBS General Purpose SSD
3	m4.large	EBS Magnetic Storage
4	m4.large	EBS General Purpose SSD

Danger

Resource Tagging and AMI ID

Remember to plan your budget accordingly for the following tasks as it will be counted for the ongoing project. If you have finished the current individual project and still have budget to spare, you can make use of the leftover budget to complete the tasks in this primer.

For this checkpoint, assign the tags with Key: `project` and Value as the tag value being used in **the currently ongoing project**. Also, keep an additional tag with Key: `Primer` and Value: `StorageBenchmarking` for all resources. For this checkpoint, provision `t3.micro` and `m4.large` instances using the AMI ID: ami-059eeca93cf09eebd (Ubuntu 16.04 Server).

Sysbench is a suite containing multiple benchmarks. You can install the sysbench using the following commands.

```
sudo apt-get update
sudo apt-get -y install sysbench
```

Your Task

Follow the steps in the next few sections to run the sysbench benchmark for each scenario. After completing your benchmark for each scenario, please note down the final **RPS (Request per Second)** reported by sysbench.

Information

Hint

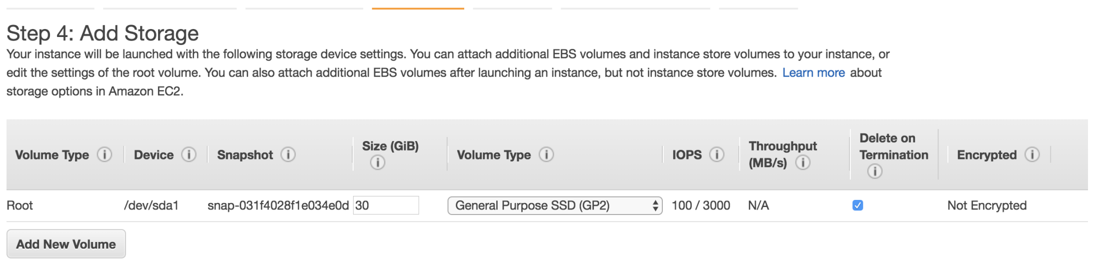
Data preparation can take a very long time if you use "slow" resources. Make wise decisions so that you don't spend a lot of time preparing the 10GB data. Be mindful of the budget as well.

Provision The Resources and Prepare Test Data

To run the benchmark for this experiment, you need to get the EC2 instances, volumes and test data ready.

Note that root volume is different from additional volume. All the experiment in this primer will be operated on the additional volumes, **NOT the root volumes**. It is mandatory to specify a root volume when you provision an EC2 instance, and the additional volumes are optional. You can either use "Add New Volume" to add additional volumes in the "Add Storage" step. Alternatively, you can provision additional EBS volumes and attach them after launching an instance.

Please note that only the root volumes and the additional volumes that are configured in the "Add Storage" with "Delete on Termination" checked will be automatically deleted when you terminate the instance. An additional volume created through the Create Volume Page (https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#CreateVolume:) will **NOT** be deleted when you terminate the instance it is attached to. Remember to terminate the additional volumes manually after you finish the project to prevent unnecessary cost.



Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. Learn more about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encrypted
Root	/dev/sda1	snap-031f4028f1e034e0d	30	General Purpose SSD (GP2)	100 / 3000	N/A	✓	Not Encrypted

1. Launch a `t3.micro` or `m4.large` instance with the AMI given above. Please keep the default configuration (the size here does not matter, as we will attach separate EBS volume for testing) at the "Add Storage" step (the same as the image above.)
2. Create a **20GB** EBS volume of the desired type (Magnetic or SSD) using the Create Volume Page (https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#CreateVolume:). Make sure the EBS volume is in the **same availability zone** as your instance.
3. Attach the EBS volume to the instance you just launched using the AWS console. You need to specify the device name of the attachment as `/dev/sd[f-p]`. **We assume that the device name is set as `/dev/sdf` in this tutorial.**

   You will get a warning as follows: "Note: Newer Linux kernels may rename your devices to `/dev/nvme[1-9]n1` internally, even when the device name entered here (and shown in the details) is `/dev/sd[f-p]` ". In the primer, when you use the instances launched using the student AMI, the renaming will take place. **Therefore, the actual device name will be `/dev/nvme1n1` instead. While in m4.large, the device name might be `/dev/xvdf`.** In the following steps, `/dev/nvme1n1` refers to the device name, **so you need to adjust this name according to the actual device name**

4. SSH into the EC2 instance.
5. List all the available drives using GNU `parted`. GNU `parted` is a program for creating, destroying, resizing, checking and copying partitions, and the file systems on them. `parted`, the partition editor, is used to manipulate the partition table on block devices. You can read more here (https://www.gnu.org/software/parted/manual/html_chapter/parted_1.html).

```
$ sudo /sbin/parted -l
Model: NVMe Device (nvme)
Disk /dev/nvme0n1: 8590MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system  Flags
 1      1049kB  8590MB  8589MB  primary  ext4         boot


Error: /dev/nvme1n1: unrecognised disk label
Model: NVMe Device (nvme)
Disk /dev/nvme1n1: 21.5GB
Sector size (logical/physical): 512B/512B
Partition Table: unknown
Disk Flags:
```

6. You get the `unrecognised disk label` error because `/dev/xvdf` is not formatted as a file system. Create an ext4 file system on `/dev/nvme1n1` with `mkfs.ext4`.

```
$ sudo /sbin/mkfs.ext4 /dev/nvme1n1
```

7. Use `parted` again to validate the volume is formatted.

```
$ sudo /sbin/parted -l
Model: NVMe Device (nvme)
Disk /dev/nvme0n1: 8590MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system  Flags
 1      1049kB  8590MB  8589MB  primary  ext4         boot


Model: NVMe Device (nvme)
Disk /dev/nvme1n1: 21.5GB
Sector size (logical/physical): 512B/512B
Partition Table: loop
Disk Flags:

Number  Start  End     Size    File system  Flags
 1      0.00B  21.5GB  21.5GB  ext4
```

8. Mount the volume to a directory. **We assume that the mounted directory is `/mnt/ebs1` in this tutorial.**

```
# make directory
sudo /bin/mkdir -p /mnt/ebs1
# mount the device to the directory
sudo /bin/mount /dev/nvme1n1 /mnt/ebs1
```

9. Validate that volume is mounted.

```
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            476M     0  476M   0% /dev
tmpfs           100M  3.4M   96M   4% /run
/dev/nvme0n1p1       7.7G  962M   6.8G  13% /
tmpfs           483M     0  483M   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           483M     0  483M   0% /sys/fs/cgroup
/dev/loop0       88M   88M     0 100% /snap/amazon-ssm-agent/495
/dev/loop1       13M   13M     0 100% /snap/core/5328
tmpfs           97M     0  97M   0% /run/user/1000
/dev/nvme1n1     20G   44M   19G   1% /mnt/ebs1 # the file system on the a
ttached volume
```

10. Change the working directory to the directory where your EBS volume was mounted.

```
cd /mnt/ebs1
```

11. Run the following command to generate test data:

```
sudo /usr/bin/sysbench --test=fileio --file-total-size=10G prepare
```

12. The above command will generate 10GB of test data on your EBS volume, you can reuse this EBS volume for all following steps.

13. It is recommended that your reboot your instance and mount the volume again if you are going to run the following experiments on the same instance that you prepared the data.

Danger

Warning

Throughout this section, please make sure all your EC2 instances are in the **same availability zone** as your EBS volume! Otherwise you won't be able to attach your EBS volume correctly!

Experiment 1 (Scenarios 1 & 2 in the table above)

To run the benchmark for this experiment, perform the following steps:

1. Launch a `t3.micro` instance with the AMI given above (you can reuse an instance if you already have one).
2. Attach and mount the EBS volume to the instance.
3. Run the following command on the data **three (3)** times (without allowing for a long delay between runs).

```
cd /mnt/ebs1
sudo /usr/bin/sysbench --test=fileio --file-total-size=10G --file-test-mod
e=rndrw --max-time=300 --max-requests=0 run
```

4. Note down the final **RPS (Request per Second)** reported by sysbench.
5. Repeat the benchmarking steps for the next storage type (with the same instance type).

Information

Note

Do not delete your EBS volume at this point, you need to reuse it for the next steps.

Experiment 2 (Scenarios 3 & 4 in the table above)

You can reuse the test files generated by the `sysbench prepare` command from the previous experiment.There is no need to format the volumes from the previous experiment, or you will lose the data.

1. Launch an `m4.large` instance with the AMI given above (you can reuse if you already have one).
2. Unmount the EBS volume if it is currently attached to the previous instance.

```
sudo /bin/umount /dev/nvme1n1
```

3. Detach the volume via the AWS console if it is currently attached to the previous instance.
4. Attach and mount the EBS volume to the instance.

5. Run the following command on the data **three (3)** times (without allowing for a long delay between runs).

```
cd /mnt/ebs1
sudo /usr/bin/sysbench --test=fileio --file-total-size=10G --file-test-mod
e=rndrw --max-time=300 --max-requests=0 run
```

6. Note down the final **RPS (Request per Second)** reported by sysbench.
7. Repeat the benchmarking steps for the next storage type (with the same instance type).

Information

Note

If you want to redo the experiments, please reboot the instance from the AWS console. After rebooting, you need to mount the disk again. But **DON'T** format the disk or generate test files using the `sudo /usr/bin/sysbench prepare` command.

Now that you have completed running the benchmark and noted down the final **RPS (Request per Second)** reported by sysbench for each of the above scenarios.

Success

Success

Congratulations on completing the Storage Benchmarking primer, this knowledge will serve you well in the future!

Danger

Remember to terminate all resources!

This includes EC2 instance, EBS volumes, and any other cloud resource that you may have created.

Do **NOT** forget to delete the EBS volumes used. You pay for each GB of the storage you provision. EBS General Purpose SSD (gp2) Volumes charges $0.10 per GB-month in `us-east-1`, and it means each 30GB volume will cost $0.1 per day. You can view the volumes using the console (https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#Volumes:sort=desc:createTime).