May 10, 2023 | CentOS Linux Questions and Answers RHEL Ubuntu

How to Check Disk Performance (IOPS and Latency) in Linux?

In this article we will discuss how to check the performance of a disk or storage array in Linux. **IOPS** (input/output operations per second) is the number of input-output operations a data storage system performs per second (it may be a single disk, a RAID array or a LUN in an external storage device). In general, IOPS refers to the number of blocks that can be read from or written to a media.

Most disk manufacturers specify nominal IOPS values, but in fact these are not guaranteed. To understand the performance of your storage subsystem prior to starting a project, it is worth getting the maximum IOPS values your storage can handle.

Using FIO (Flexible I/O) Tool for Storage Benchmarking

To measure disk IOPS performance in Linux, you can use the **fio** (the tool is available for CentOS/RHEL in EPEL repository). So, to install fio in RHEL or CentOS, use the yum (dnf) package manager:

```
# yum install epel-release -y
# yum install fio -y
```

Or apt-get in Debian or Ubuntu:

```
# apt-get install fio
```

```
Installing : libpmem-1.5.1-2.1.el7.x86_64
Installing : daxctl-libs-64.1-2.el7.x86_64
Installing : ndctl-libs-64.1-2.el7.x86_64
Installing : libpmemblk-1.5.1-2.1.el7.x86_64
Installing : pciutils-3.5.1-3.el7.x86_64
Installing : rdma-core-22.1-3.el7.x86_64
Installing : libibverbs-22.1-3.el7.x86_64
Installing : librdmacm-22.1-3.el7.x86_64
Installing : fio-3.7-1.el7.x86_64
Verifying  : daxctl-libs-64.1-2.el7.x86_64
Verifying  : libibverbs-22.1-3.el7.x86_64
Verifying  : fio-3.7-1.el7.x86_64
Verifying  : libpmemblk-1.5.1-2.1.el7.x86_64
Verifying  : librdmacm-22.1-3.el7.x86_64
Verifying  : pciutils-3.5.1-3.el7.x86_64
Verifying  : rdma-core-22.1-3.el7.x86_64
Verifying  : libpmem-1.5.1-2.1.el7.x86_64
Verifying  : ndctl-libs-64.1-2.el7.x86_64
```

Then you to identify the disks to test. The test is done by performing read/write operations in the directory your disk or LUN is mounted to.

Let's do several types of disk IOPS performance tests in various disk load scenarios (a test mode you select depending on a hosted app logic and general infrastructure of a project).

Random Read/Write Operation Test

When running the test, an 8 GB file will be created. Then fio will read/write a **4KB** block (a standard block size) with the **75/25%** by the number of reads and writes operations and measure the performance. The command is as follows:

```
# fio --randrepeat=1 --ioengine=libaio --direct=1 --gtod_reduce=1 --name=fiotest --filename=testfio --bs=4k --iodepth=64 --size=8G --readwrite=randrw --rwmixread=75
```

```
Starting 1 process
fiotest: Laying out IO file (1 file / 8192MiB)
Jobs: 1 (f=1): [m(1)][100.0%][r=330MiB/s,w=111MiB/s][r=84.5k,w=28.4k IOPS][eta 00m:00s]
fiotest: (groupid=0, jobs=1): err= 0: pid=14989: Wed Apr 15 11:26:19 2020
   read: IOPS=83.0k, BW=328MiB/s (344MB/s)(6141MiB/18721msec)
    bw (  KiB/s): min=325984, max=340576, per=100.00%, avg=335975.14, stdev=2743.98, samples=37
    iops        : min=81496, max=85144, avg=83993.84, stdev=686.08, samples=37
  write: IOPS=28.0k, BW=110MiB/s (115MB/s)(2051MiB/18721msec)
    bw (  KiB/s): min=108792, max=115024, per=100.00%, avg=112183.57, stdev=1124.85, samples=37
    iops        : min=27198, max=28756, avg=28045.89, stdev=281.21, samples=37
  cpu           : usr=10.64%, sys=84.80%, ctx=47775, majf=0, minf=27
  IO depths     : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.1%, 32=0.1%, >=64=100.0%
     submit     : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     complete   : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.1%, >=64=0.0%
     issued rwts: total=1572145,525007,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency    : target=0, window=0, percentile=100.00%, depth=64
```

I ran my first test on an array that consisted of two **SSD**s and got good results:

- **Read:** 3280MiB/s, IOPS avg 83000
- **Write:** 110MiB/s, IOPS avg 28000

Since we have run a combined read/write test, the values for the separate tests will be higher.

In comparison, I measured the performance on a **SATA** drive:

```
fio-3.7
Starting 1 process
fiotest: Laying out IO file (1 file / 8192MiB)
Jobs: 1 (f=1): [m(1)][100.0%][r=2746KiB/s,w=940KiB/s][r=686,w=235 IOPS]
fiotest: (groupid=0, jobs=1): err= 0: pid=15329: Wed Apr 15 12:31:11 20
   read: IOPS=430, BW=1724KiB/s (1765kB/s)(6141MiB/3648426msec)
    bw (  KiB/s): min=  368, max= 3448, per=100.00%, avg=1723.21, stdev=
    iops        : min=   92, max=  862, avg=430.79, stdev=79.06, samples
  write: IOPS=143, BW=576KiB/s (589kB/s)(2051MiB/3648426msec)
    bw (  KiB/s): min=   32, max= 1117, per=100.00%, avg=575.41, stdev=1
    iops        : min=    8, max=  279, avg=143.81, stdev=32.57, samples
  cpu           : usr=1.42%, sys=3.74%, ctx=1926680, majf=0, minf=26
```

- **Read:** IOPS=430, BW=1.7 MiB/s

- **Write:** IOPS=143, BW= 0.6 MiB/s

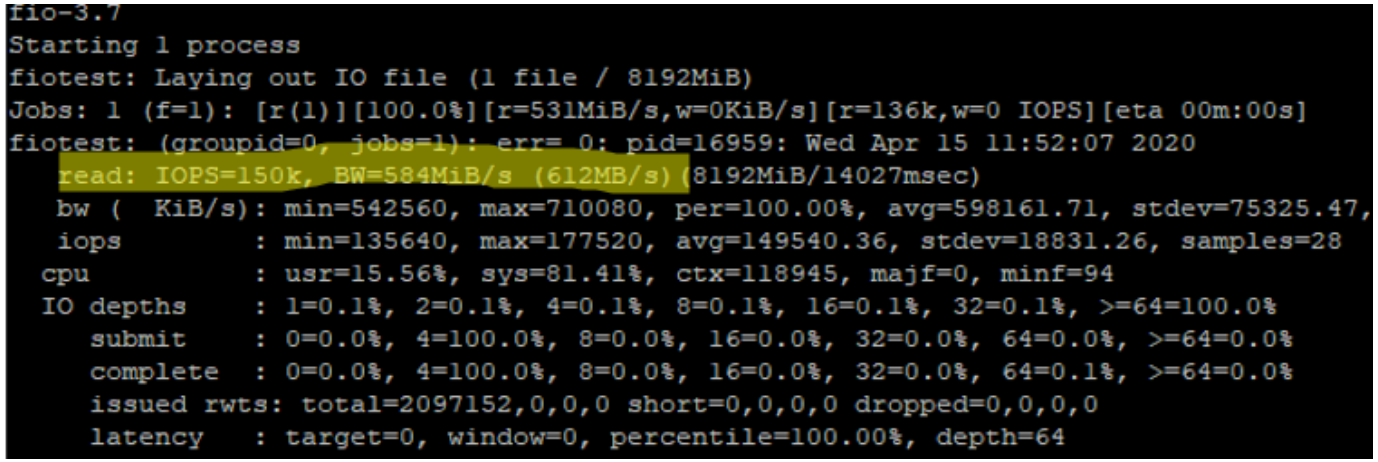
Of course, the HDD results are worse than those of the SSD.

Random Read Operation Test

To measure disk performance for random read operations only, run the following command:

```
# fio --randrepeat=1 --ioengine=libaio --direct=1 --gtod_reduce=1 --name=fiotest --filename=testfio --bs=4k --iodepth=64 --size=8G --readwrite=randread
```

The final part of the command was changed to `--readwrite=randread`.



As I told earlier, the read/write performance will be higher if measured separately:

```
READ: IOPS=150k, BW=584MiB/s (612MB/s)
```

Random Write Operation Test

To measure disk performance for random write operations, run this command:

```
# fio --randrepeat=1 --ioengine=libaio --direct=1 --gtod_reduce=1 --name=fiotest --filename=fiotest --bs=4k --iodepth=64 --size=8G --readwrite=randwrite
```

```
WRITE: IOPS=84.7k, BW=331MiB/s (347MB/s)
```

Write operation performance on good SSDs is also very high. Like in read operation test, the difference as compared to a mixed test reaches 200-250 MiB/s and 50000 IOPS.

If you refer to the official manufacturer documentation (these are Intel SSDs), it is safe to say that the values are true.

Fio Config File Examples

Fio allows to check disk performance using interactive commands and with configuration files prepared in advance for testing. To use the this option, create a file:

```
# nano read.fio
```

And add the following contents into it:

```
[global]
rw=randread
size=8G
filename=/tmp/testfio
ioengine=libaio
iodepth=4
invalidate=1
direct=1
[bgread]
rw=randread
iodepth=64
```

Then start the test:

```
# fio read.fio
```

The test will measure the read performance of a disk. To test write performance, use the following config file:

```
[global]
rw=randwrite
size=8G
filename=/tmp/testfio
ioengine=libaio
iodepth=4
```

```
invalidate=1
direct=1
[bgwrite]
rw=randwrite
iodepth=64
```

Measuring Disk Latency Using Ioping

Besides IOPS, there is another important parameter that characterizes the quality of your storage: it is latency. **Latency** is an input/output request delay that determines the time of access to a storage (measured in milliseconds). The higher the latency is, the more your app has to wait till it gets data from your disk. The latency values **over 20 ms** for typical data storage systems are considered poor.

To check disk latency in Linux, the **ioping** tool is used:

```
# yum install ioping -y
```

```
# apt-get install ioping
```

Run the latency test for your disk (20 requests are run):

```
# ioping -c 20 /tmp/
```

```
4 KiB <<< /tmp/ (ext4 /dev/md126p5): request=1 time=1.55 ms (warmup)
.....................
4 KiB <<< /tmp/ (ext4 /dev/md126p5): request=19 time=176.3 us (fast)
4 KiB <<< /tmp/ (ext4 /dev/md126p5): request=20 time=356.9 us
--- /tmp/ (ext4 /dev/md126p5) ioping statistics ---
19 requests completed in 5.67 ms, 76 KiB read, 3.35 k iops, 13.1 MiB/s
generated 20 requests in 19.0 s, 80 KiB, 1 iops, 4.21 KiB/s
min/avg/max/mdev = 176.3 us / 298.7 us / 368.9 us / 45.7 us
```

The average value is **298.7 us** (microseconds), so the average latency in our case is **0.3 ms**, that is excellent.

> The latency value can be specified in **us** (microseconds) or **ms** (milliseconds). To get a ms value from an us one, divide it by 1,000.

So you can perform a storage load test on your server prior to launching a project and check the highest performance values. However, the test doesn't guarantee that your disk array or disk will show the same performance constantly, but it is worth to take the test on the initial stage of a project. Learn how to test IOPS in Windows in this article.

对 EBS 卷进行基准测试

PDF (ec2-ug.pdf#benchmark_procedures) | RSS (amazon-ec2-release-notes.rss)

您可以通过模拟 I/O 工作负载来测试 Amazon EBS 卷的性能。过程如下所述：

1. 启动 EBS 优化实例。
2. 创建新的 EBS 卷。
3. 将这些卷附加到您的 EBS 优化实例。
4. 配置并挂载块储存设备。
5. 安装工具以便测试 I/O 性能。
6. 测试卷的 I/O 性能。
7. 删除卷并终止实例，确保不会继续引发更改。

> ⚠ **重要**
>
> 某些过程可能会对您进行基准测试的 EBS 卷上的现有数据造成破坏。基准测试程序适用于出于测试目的而特别创建的卷，并不适用于生产卷。

设置实例

为了获得最佳的 EBS 卷性能，我们建议您使用 EBS 优化实例。EBS 优化实例可在 Amazon EC2 和 Amazon EBS 之间提供实例专用吞吐量。EBS 优化的实例在 Amazon EC2 与 Amazon EBS 之间提供了专用带宽，其规格取决于实例类型。有关更多信息，请参阅 Amazon EBS 优化的实例 (./ebs-optimized.html) 。

要创建 EBS 优化实例，可在使用 Amazon EC2 控制台启动实例时选择**作为 EBS 优化的实例启动**，或在使用命令行时指定 **--ebs-optimized**。请确保您启动的实例是支持该选项的最新一代实例。有关更多信息，请参阅Amazon EBS 优化的实例 (./ebs-optimized.html) 。

设置 Provisioned IOPS SSD 或 通用型 SSD 卷

要使用 Amazon EC2 控制台创建预置 IOPS SSD（`io1` 和 `io2`）或通用型 SSD（`gp2` 和 `gp3`）卷，对于 **Volume type (卷类型)**，选择 **Provisioned IOPS SSD (io1) (预置 IOPS SSD [io1])**、**Provisioned IOPS SSD (io2) (预置 IOPS SSD [io2])**、**General Purpose SSD (gp2) (通用型 SSD [gp2])**，或 **General Purpose SSD (gp3) (通用型 SSD [gp3])**。在命令行中，为 `io1` 参数指定 `io2`、`gp2`、`gp3` 或 **--volume-type**。对于 `io1`、`io2` 和 `gp3` 卷，请指定 **--iops** 参数的每秒 I/O 操作数 (IOPS)。有关更多信息，请参阅Amazon EBS 卷类型 (./ebs-volume-types.html) 和创建 Amazon EBS 卷 (./ebs-creating-volume.html) 。

要了解这些示例测试，我们建议您创建一个包含 6 个卷的高性能 RAID 0 阵列。因为您是按照预配置的 GB 数量（以及为 io1、io2 和 gp3 卷预配置的 IOPS 数量，而不是卷的数量）付费，因此创建多个较小卷并使用它们来创建条带集不会产生额外费用。如果您是使用 Oracle Orion 来测试卷的性能，则它可以模拟 Oracle ASM 的条带化操作，因此我们建议您让 Orion 执行条带化分区。如果您使用的是其他基准测试工具，则需要自己对卷执行条带化分区。

有关如何创建包含 6 个卷的 RAID 0 阵列的说明，请参阅 在 Linux 上创建 RAID 0 阵列 (./raid-config.html#linux-raid) 。

设置吞吐量优化型 HDD（st1）卷或 Cold HDD（sc1）卷

要创建 `st1` 卷，可在使用 Amazon EC2 控制台创建卷时选择 **Throughput Optimized HDD (吞吐量优化型 HDD)**，或在使用命令行时指定 **--type st1**。要创建 `sc1` 卷，可在使用 Amazon EC2 控制台创建卷时选择 Cold HDD，或在使用命令行时指定 **--type sc1**。有关创建 EBS 卷的信息，请参阅创建 Amazon EBS 卷 (./ebs-creating-volume.html) 。有关将这些卷附加到您的实例的信息，请参阅 将 Amazon EBS 卷挂载到实例 (./ebs-attaching-volume.html) 。

AWS 提供了一个 JSON 模板，以便与 AWS CloudFormation 配合使用来简化此设置过程。访问模板 (https://s3.amazonaws.com/cloudformation-examples/community/st1_cloudformation_template.json) 并将其另存为 JSON 文件。AWS

CloudFormation 允许您配置自己的 SSH 密钥并提供了更简单的方式来设置性能测试环境，以评估 `st1` 卷。此模板会创建一个最新一代的实例以及一个 2 TiB 的 `st1` 卷，然后将该卷附加到 `/dev/xvdf` 处的实例。

使用模板创建 HDD 卷

1. 打开 AWS CloudFormation 控制台，地址：https://console.aws.amazon.com/cloudformation (https://console.aws.amazon.com/cloudformation/) 。
2. 选择 **Create Stack**。
3. 选择 **Upload a Template to Amazon S3**，然后选择之前获得的 JSON 模板。
4. 为您的堆栈提供名称 (如“ebs-perf-testing”)，然后选择实例类型 (默认为 r3.8xlarge) 和 SSH 密钥。
5. 选择 **Next** 两次，然后选择 **Create Stack**。
6. 新堆栈的状态从 **CREATE_IN_PROGRESS** 变为 **COMPLETE** 后，选择 **Outputs**（输出）以获取新实例的公有 DNS 条目，新实例将附加一个 2TiB 的 `st1` 卷。
7. 以用户 **`ec2-user`** 的身份使用 SSH 连接到您的新堆栈（使用从上一步的 DNS 条目中获得的主机名）。
8. 继续执行安装基准测试工具 (#install_tools) 。

安装基准测试工具

下表列出了您可用于对 EBS 卷的性能进行基准测试的部分可用工具。

工具	描述
fio	用于测试 I/O 性能。（请注意，**fio** 依赖于 `libaio-devel`。） 要在 Amazon Linux 上安装 **fio**，请运行以下命令： `[ec2-user ~]$ sudo yum install`

工具	描述

	`-y fio`
	
	要在 Ubuntu 上安装 **fio**，请执行以下命令：
	
	`sudo apt-get install -y fio`

Oracle Orion 校准工具 (https://docs.oracle.com/cd/E18283_01/server.112/e16638/iodesign.htm#BABFCFBC)	用于校准要与 Oracle 数据库搭配使用的存储系统的 I/O 性能。

这些基准测试工具可支持各种测试参数。您应该使用命令来测试您的卷支持的工作负载。下面提供的命令示例可帮助您入门。

选择卷队列长度

基于工作负载和卷类型选择最佳卷队列长度。

SSD 支持的卷的队列长度

要确定支持 SSD 的卷上工作负载的最佳队列长度，建议您将每 1000 IOPS（通用型 SSD 卷的基准量，Provisioned IOPS SSD 卷的预置量）对应 1 个队列长度作为目标。然后，您可以监控应用程序性能，并根据应用程序需求调整该值。

在达到预配置 IOPS、吞吐量或最佳系统队列长度值之前，增加队列长度有好处，当前队列长度设置为 32。举例来说，预配置 3,000 IOPS 的卷应该将队列长度设置为 3。您应该尝试将这些值调高或调低，看看对于您的应用程序，什么样的设置能够实现最佳性能。

HDD 支持的卷的队列长度

要确定 HDD 卷上工作负载的最佳队列长度，建议您在执行 1MiB 顺序 I/O 时以至少为 4 的队列长度作为目标。然后，您可以监控应用程序性能，并根据应用程序需求调整该值。例如，突发吞吐量为 500 MiB/s、IOPS 为 500 的 2 TiB `st1` 卷在执行 1024 KiB、512 KiB

或 256 KiB 的顺序 I/O 时，分别应该将队列长度 4、8 或 16 作为目标。您应该尝试将这些值调高或调低，看看对于您的应用程序，什么样的设置能够实现最佳性能。

禁用 C 状态

在运行基准测试之前，您应禁用处理器 C 状态。支持此功能的 CPU 中的核心在暂时空闲时，会进入 C 状态以节省功耗。在调用核心以恢复处理时，将经过一段特定的时间，核心才能再次全速运行。此延迟可能会干扰处理器基准测试例程。有关 C 状态以及哪些 EC2 实例类型支持此状态的更多信息，请参阅 EC2 实例的处理器状态控制 (https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/processor_state_control.html) 。

在 Linux 上禁用 C 状态

您可在 Amazon Linux、RHEL 和 CentOS 上按以下所示禁用 C 状态：

1. 获取 C 状态数。

```
$ cpupower idle-info | grep "Number of idle states:"
```

2. 从 c1 到 cN 禁用 C 状态。理想情况下，核心应处于状态 c0。

```
$ for i in `seq 1 $((N-1))`; do cpupower idle-set -d $i; done
```

执行基准测试

以下步骤介绍各种 EBS 卷类型的基准测试命令。

对附加了 EBS 卷的 EBS 优化实例运行以下命令。如果已从快照创建 EBS 卷，在执行基准测试之前，请确保初始化这些卷。有关更多信息，请参阅初始化 Amazon EBS 卷 (./ebs-initialize.html) 。

完成对卷的测试后，可参阅以下主题来帮助清除卷：删除 Amazon EBS 卷 (./ebs-deleting-volume.html) 和终止实例 (./terminating-instances.html) 。

基准 Provisioned IOPS SSD 和 通用型 SSD 卷

在您创建的 RAID 0 阵列上运行 **fio**。

以下命令可执行 16 KB 随机写入操作。

```
[ec2-user ~]$ sudo fio --directory=/mnt/p_iops_vol0 --ioengine=psync --name fio_test_file --
direct=1 --rw=randwrite --bs=16k --size=1G --numjobs=16 --time_based --runtime=180 --
group_reporting --norandommap
```

以下命令可执行 16 KB 随机读取操作。

```
[ec2-user ~]$ sudo fio --directory=/mnt/p_iops_vol0 --name fio_test_file --direct=1 --rw=randread -
-bs=16k --size=1G --numjobs=16 --time_based --runtime=180 --group_reporting --norandommap
```

有关解析结果的更多信息，请参阅以下教程：使用 fio 检查磁盘 IO 性能 (https://www.linux.com/tutorials/inspecting-disk-io-performance-fio/) 。

基准 st1 和 sc1 卷

在 **fio** 或 `st1` 卷上运行 `sc1`。

> **注意**
>
> 在执行这些测试之前，请按为 st1 和 sc1 上的高吞吐量读取密集型工作负载增加预读值 (./EBSPerformance.html#read_ahead) 所述在实例上设置缓冲 I/O。

以下命令针对附加的 `st1` 块储存设备（例如 `/dev/xvdf`）执行 1 MiB 的顺序读取操作：

```
[ec2-user ~]$ sudo fio --filename=/dev/<device> --direct=1 --rw=read --randrepeat=0 --
ioengine=libaio --bs=1024k --iodepth=8 --time_based=1 --runtime=180 --name=fio_direct_read_test
```

以下命令针对附加的 st1 块储存设备执行 1 MiB 的顺序写入操作：

```
[ec2-user ~]$ sudo fio --filename=/dev/<device> --direct=1 --rw=write --randrepeat=0 --
ioengine=libaio --bs=1024k --iodepth=8 --time_based=1 --runtime=180 --name=fio_direct_write_test
```

有些工作负载可对块储存设备的不同部分混合执行顺序读取和顺序写入操作。要对此类工作负载进行基准测试，我们建议您为读取和写入操作单独、同时使用 **fio** 作业，并为每个作业使用 **fio** offset_increment 选项将块储存设备的不同位置作为目标。

运行此类工作负载比顺序写入或顺序读取工作负载要复杂一些。使用文本编辑器创建一个 fio 作业文件，在此示例中名为 fio_rw_mix.cfg，包含以下内容：

```
[global]
clocksource=clock_gettime
randrepeat=0
runtime=180

[sequential-write]
bs=1M
ioengine=libaio
direct=1
iodepth=8
filename=/dev/<device>
do_verify=0
rw=write
rwmixread=0
rwmixwrite=100

[sequential-read]
```

```
bs=1M
ioengine=libaio
direct=1
iodepth=8
filename=/dev/<device>
do_verify=0
rw=read
rwmixread=100
rwmixwrite=0
offset=100g
```

然后运行以下命令：

```
[ec2-user ~]$ sudo fio fio_rw_mix.cfg
```

有关解析结果的更多信息，请参阅以下教程：使用 fio 检查磁盘 IO 性能 (https://www.linux.com/tutorials/inspecting-disk-io-performance-fio/) 。

对于 **fio** 和 `st1` 卷而言，通过多个 `sc1` 作业来执行直接 I/O（即使使用顺序读入或写入操作）可能会导致吞吐量小于预期数值。建议您使用一个直接 I/O 作业并使用 `iodepth` 参数来控制并发 I/O 操作的数量。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。

测试磁盘性能

PDF (sql-server-ec2-best-practices.pdf#disk-perf) | RSS (sql-server-ec2-best-practices.rss)

我们建议您使用类似的工具检查磁盘性能DiskSpd (https://github.com/microsoft/diskspd) 。在运行 SQL Server 特定测试之前，此工具可以估算出磁盘速度。评估磁盘性能非常重要，因为 EBS 卷的工作方式与本地环境中的传统 SAN 不同。缺少适当的性能测试可能会导致迁移后的性能意外降低。你也可以跑自定义测试 (https://github.com/Microsoft/diskspd/wiki/Customizing-tests) 和 DiskSpd。

r/linuxadmin • 2 yr. ago
[deleted]

Join

What Linux tools do you use to check IOPS

Hello,

I was wondering what command/tool you use on your Linux devices to show disk performance/IOPS?

Plus what values might be deemed good and bad?

Thanks

61 23 Share

Sort by: Best

Add a Comment

b_moldo • 2 yr. ago

Iostat

- https://coderwall.com/p/utc42q/understanding-iostat
- https://www.igvita.com/2009/06/23/measuring-optimizing-io-performance/

Fio

- https://tobert.github.io/post/2014-04-28-getting-started-with-fio.html
- https://tobert.github.io/post/2014-04-17-fio-output-explained.html
- http://serverfault.com/questions/677340/poor-iscsi-performance-with-ssd-disks-and-10-gbe-network

DD

- simple sequential I/O performance measurements

Bonnie++

- For random tests
- Attention: bonnie++ creates an enourmous read and write queue thus the load average will increase to 15+
- https://www.jamescoyle.net/how-to/599-benchmark-disk-io-with-dd-and-bonnie

33 Reply Share ...

ESCAPE_PLANET_X • 2 yr. ago

Beware IO benchmarking tools like FIO. It can be used to get useful information it can also generate very misleading results.

CPU bottlenecks, FS magic tricks, etc.

9 Reply Share ...

2 more replies

1 more reply

ztherion • 2 yr. ago

bcc includes a number of example scripts such as biolatency and disksnoop. See the tools dir at https://github.com/iovisor/bcc. Depending on your distro these may be available as a package.

see also https://www.brendangregg.com/ebpf.html

15 Reply Share ...

pino_entre_palmeras • 2 yr. ago • Edited 2 yr. ago

~~Can~~ Can't stress Gregg's writing enough.

You can also start with the storage section here:

https://www.brendangregg.com/USEmethod/use-linux.html

Edit: Typo

12 Reply Share ...

2 more replies

[deleted] • 2 yr. ago

Note that on Debian distros you are looking for bpfcc which won't necessarily show up if you search for bcc.

4 Reply Share ...

[deleted] • 2 yr. ago

CoaxVex • 2 yr. ago

iotop

9 Reply Share ...

1 more reply

mikelieman • 2 yr. ago

Are we not using 'sar' anymore?

4 Reply Share ...

1 more reply

snark42 • 2 yr. ago

I use telegraf to collect stats and send to InfluxDB. Grafana to view stats. Of course this includes a lot more than just IO/IOPS.

3 Reply Share ...

4 more replies

tkyjonathan • 2 yr. ago

iotop

3 Reply Share ...

mgedmin • 2 yr. ago

`atop` or `iotop`.

I like that `atop` shows disk utilization in %, which makes it easy to see when things go bad (100% busy means a bottleneck on disk i/o).

2 Reply Share ...

[deleted] • 2 yr. ago

wow there is a lot in atop to see, what column do you normally use to check IOPS?

1 Reply Share ...