# 在Linux中使用Azure Premium 存储的基本优化指南

注：以下测试和结果都是基于CentOS 6.5。对于其他版本，请参考本文档，并自行进行相关测试。

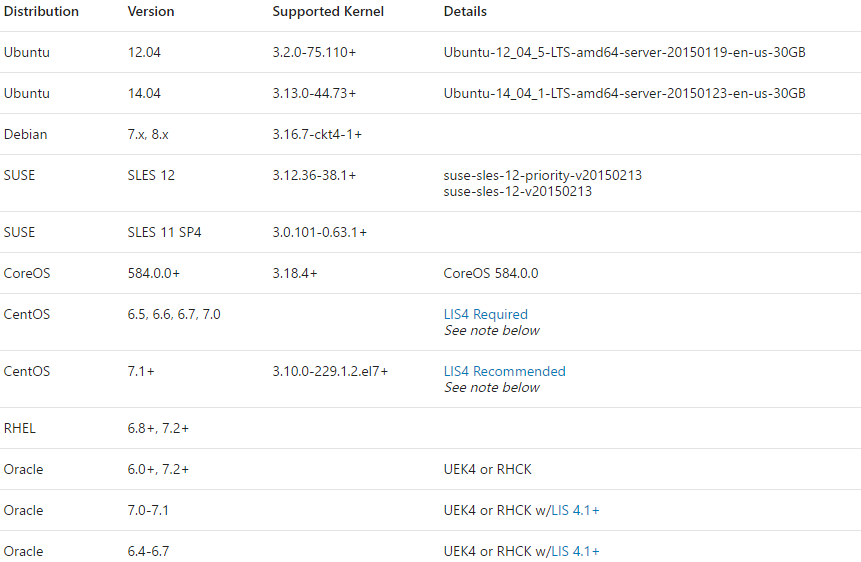
1. 建议使用最新的内核版本

一般情况下，新的内核版本能解决老版本中存在的问题，添加对新出现硬件的支持，以及改进对硬件的支持程度。

建议使用你当前Linux发行版本中最新的内核版本；如有可能，定期更新到大版本中最新的补丁版本。

1. 更新Linux LIS驱动

官方文档中提到，CentOS和Oracle 特定的发行版本，需要更新LIS以支持Premium storage。具体如下图所示：



<https://docs.microsoft.com/en-us/azure/storage/storage-premium-storage>

1. 在磁盘分区时，与设备的页大小对齐

在2.6.32-431.29.2.el6.x86\_64的内核中，fdisk默认还是使用柱面为边界来分区；如果柱面与SSD磁盘的页大小没有对齐，文件系统层面的一个IO，可能会在设备层产生额外的开销，影响读写响应时间。

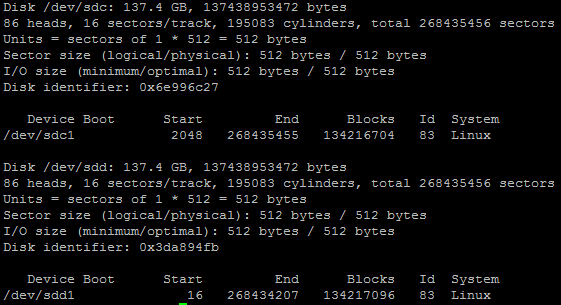
以下是CentOS 6.5执行fdisk 不带参数时的警告信息。



以下是通过不同参数进行磁盘分区后，使用fdisk -u -c -l的结果。/dev/sdc1是从磁盘的第二个1024k开始的，/dev/sdd1则是从第二个8k开始的。一般来讲，这个边界对齐最好是128K的偶数倍。

#fdisk -u -c /dev/sdc

#fdisk /dev/sdd



在CentOS 7以后，fdisk默认就使用了以512Bytes的Sector大小为单位来显示，分区时也默认从第2048个sector开始划第一个分区。所以不必担心SSD还会有边界对齐问题，使用默认命令fdisk即可。

<https://wiki.archlinux.org/index.php/Improving_performance#Storage_devices>

1. 文件系统格式

在CentOS 7以后，默认使用XFS作为文件系统格式。在CentOS 6中，推荐使用ext4或者XFS。

1. 挂载选项

Write barrier 是用来保证内存中数据以正确顺序写入磁盘，以防止突然断电而引起的数据不一致问题。如果在新建磁盘时设置的cache规则为none或者readonly, 完全可以禁用该功能，因为数据会不经过cache直接写入Azure Premium Storage 底层的磁盘。

noatime/relatime: atime是文件属性中的访问时间。每次文件被访问，该时间就会被更改。如果文件被频繁访问，那花费在记录该时间上的时间也很可观，会影响IO的效率。

如果应用不需要比较文件访问时间，建议将其禁止；若因为某些原因要保持atime更新，可以设置relatime，将atime设置成和mtime一起更新。

*注：在Azure 提供的CentOS 7版本以后，默认挂载选项已经设置成relatime。*

操作方法：

命令行中使用

#mount -o noatime,barrier=0 /dev/sdxn /mount/point

或者

#mount -o relatime,barrier=0 /dev/sdxn /mount/point

在挂载配置文件中更改

#vi /etc/fstab

/dev/sdxn /mount/point ext4 noatime,barrier=0 0 0

或者

/dev/sdxn /mount/point ext4 relatime,barrier=0 0 0

Discard: 用来决定是否在有数据被删除时对SSD block进行整理，以备后续数据写入时提高速度；但如果小文件删除频繁，discard本身可能引起性能下降。慎用之。

参考：

<https://patrick-nagel.net/blog/archives/337>

<http://people.redhat.com/lczerner/discard/ext4_discard.html>

1. IO调度算法

CFQ: CentOS 6中的默认IO算法。特点是对所有有IO请求的进程，提供公平的磁盘IO带宽。

NOOP: 即 none operation。对所有的IO请求，不做任何操作，直接发给磁盘控制器。

Deadline: CentOS 7中默认IO算法。其特点在于尽量将所有的IO请求的延迟保持在一定范围内，避免有的IO一直等待无法获取响应时间。

根据应用的IO类型，选择合适的IO调度算法，能在一定程度上提供应用性能。这里讲的IO调度算法的主要目的是为了节省磁盘寻道时间。而SSD 磁盘和HDD最大的不同，在于它没有寻道时间和旋转延迟。因此对于HDD性能有极大帮助的CFQ和Deadline，以及nr\_requests/queue\_depth参数，对于SSD来讲意义不大；相反，使用noop 或deadline这种基本不排序IO的算法，反倒能提升SSD的响应时间。

Deadline 和 noop的算法还是有所不同的，建议两种分别测试比较一下，择优而用。另外，IO调度算法是根据底层磁盘来的，如果有多个磁盘，要分别进行设置。

查看IO算法：

#cat /sys/block/sdx/queue/scheduler

在线更改算法

#echo noop > /sys/block/sdx/queue/scheduler

将算法更新到启动参数中（CentOS 6）:

#vi /etc/grub.conf

kernel /boot/vmlinuz-2.6.32-431.29.2.el6.x86\_64 ro root=UUID=8fc4d768-29cd-462c-a7ab-5bf4bcfa9fa2 rd\_NO\_LUKS KEYBOARDTYPE=pc KEYTABLE=us LANG=en\_US.UTF-8 numa=off console=ttyS0,115200n8 earlyprintk=ttyS0,115200 rootdelay=300 rd\_NO\_MD SYSFONT=latarcyrheb-sun16 rd\_NO\_LVM rd\_NO\_DM elevator=deadline

使用sysbench 0.5，在1 core, 3.5 GB memory 的DS1型号的CentOS 6.5中进行了文件读写测试。经过比较，使用了noatime和barrier=0，且使用了边界对齐分区方法的ext4文件系统，在使用deadline或者noop IO算法时，平均IO时间和最大IO时间都有所下降，文件读写IO综合情况表现较好。

上述情况仅从基本操作上进行了一些优化。在实际环境中，各个参数的调整，都需要进行反复测试，以求达到最佳状态。

更多参考：

<https://blogs.msdn.microsoft.com/igorpag/2014/10/23/azure-storage-secrets-and-linux-io-optimizations/>

<https://docs.microsoft.com/en-us/azure/virtual-machines/virtual-machines-linux-classic-optimize-mysql>

<https://www.azure.cn/documentation/articles/storage-premium-storage/>

<https://access.redhat.com/solutions/32376>