鲲鹏 BoostKit 分布式存储使能套件

用户指南

文档版本 06

发布日期 2021-05-26

华为技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目 录

1 Bcache 用户指南（CentOS 7.6）

1.1 介绍

Bcache是Linux内核块层cache，它使用SSD来作为HDD硬盘的cache，从而起到加速作用。Bcache内核模块仅在Linux 3.10及以上版本支持，因此使用Bcache，需要将内核升级到3.10及以上版本，并在内核配置项中打开Bcache模块。

为了方便Bcache的使用，本文提供指导流程，可分别制作：

- Bcache模块内核RPM包
- Bcache-tools RPM包
- 包含Bcache内核RPM包、Bcache-tools RPM包的iso镜像

更多详情见Bcache主页和使用手册：

https://bcache.evilpiepirate.org/

https://evilpiepirate.org/git/linux-bcache.git/tree/Documentation/bcache.txt

1.2 环境准备

环境要求

环境要求如下表所示。

服务器名称	TaiShan 200服务器（型号2280）
OS	CentOS Linux release 7.6.1810

说明

- 由于制作安装包过程中需要在线安装依赖包，请确保服务器可以接入互联网或已配置本地源。
- 本文档提供基于CentOS 7.6的4.14内核版本的指导流程，其他版本可参考本文档执行。
- 请确保服务器安装OS与要修改OS一致。

1.3 编译内核

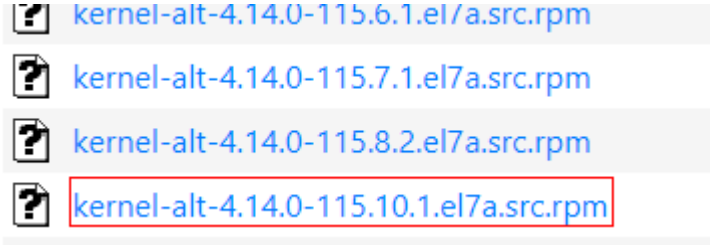
1.3.1 获取源码

步骤1 创建目录并进入该目录。

```
mkdir -p /home/kernel-bcache
cd /home/kernel-bcache
```

步骤2 下载内核源码包至“/home/kernel-bcache”目录。

下载路径：**http://vault.centos.org/7.6.1810/updates/Source/SPackages/**



步骤3 解压源码包。

```
rpm2cpio kernel-alt-4.14.0-115.10.1.el7a.src.rpm | cpio -divm
tar -xvf linux-4.14.0-115.10.1.el7a.tar.xz
```

步骤4 安装依赖。

```
yum -y install rpm-build m4 gcc xmlto asciidoc openssl-devel hmaccalc python-devel newt-devel perl-ExtUtils-Embed elfutils-devel zlib-devel binutils-devel bison audit-libs-devel java-devel numactl-devel pciutils-devel ncurses-devel createrepo genisoimage net-tools git bc
```

----结束

1.3.2 修改内核配置

步骤1 获取内核配置文件。

```
cd /home/kernel-bcache/linux-4.14.0-115.10.1.el7a
cp ../kernel-alt-4.14.0-aarch64.config .config
```

步骤2 配置Bcache模块。

```
make menuconfig
```

1. 选择“Device Drivers ”。


```
    General setup  --->
[*] Enable loadable module support  --->
-*- Enable the block layer  --->
    Platform selection  --->
    Bus support  --->
    Kernel Features  --->
    Boot options  --->
    Userspace binary formats  --->
    Power management options  --->
    CPU Power Management  --->
[*] Networking support  --->
    Device Drivers  --->
```

2. 按“Enter”键进入下一级菜单，选择“Multiple device driver support (RAID and LVM)”。

```
[*] Block devices  --->
<M> NVM Express block device
<M> NVM Express over Fabrics RDMA host driver
< > NVM Express over Fabrics FC host driver
<M> NVMe Target support
<M>   NVMe loopback device support
<M>   NVMe over Fabrics RDMA target support
< >   NVMe over Fabrics FC target driver
    Misc devices  --->
    SCSI device support  --->
<*> Serial ATA and Parallel ATA drivers (libata)  --->
[*] Multiple devices driver support (RAID and LVM)  --->
<M> Generic Target Core Mod (TCM) and ConfigFS Infrastructure  --->
```

3. 按“Enter”键进入下一级菜单，选中“Block device as cache”，键盘输入“M”选中该配置。

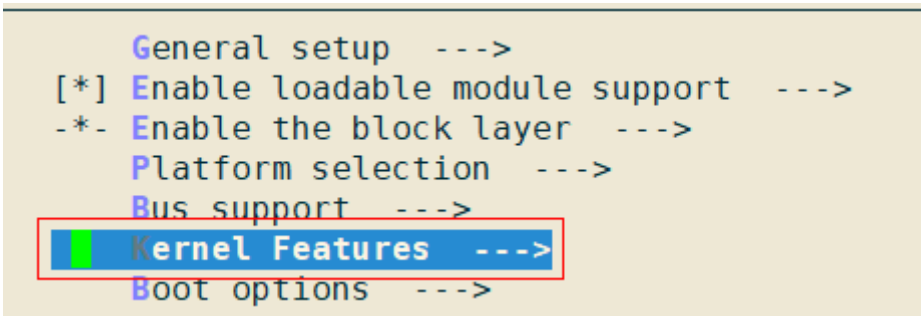
```
--- Multiple devices driver support (RAID and LVM)
{*}   RAID support
[*]     Autodetect RAID arrays during kernel boot
<M>     Linear (append) mode
{M}     RAID-0 (striping) mode
{M}     RAID-1 (mirroring) mode
{M}     RAID-10 (mirrored striping) mode
{M}     RAID-4/RAID-5/RAID-6 mode
< >     Multipath I/O support
<M>     Faulty test module for MD
<M>     Block device as cache
[ ]     Bcache debugging
```

4. 按两次“exit”返回至第一层。

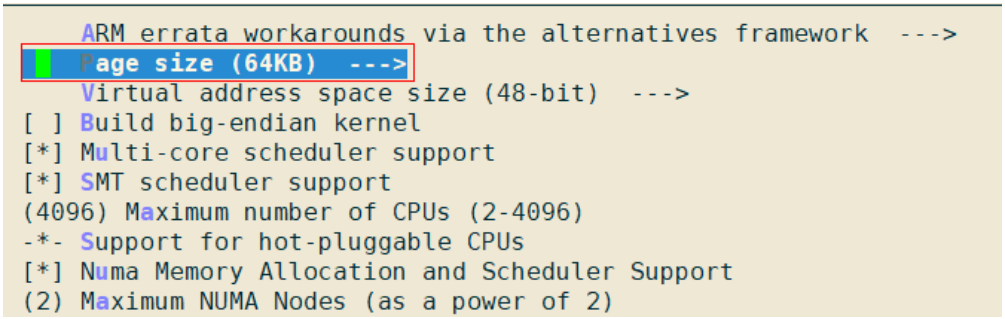
```
    <Select>   < Exit >   < Help >   < Save >   < Load >
```

步骤3 修改内核PAGESIZE大小为4K。

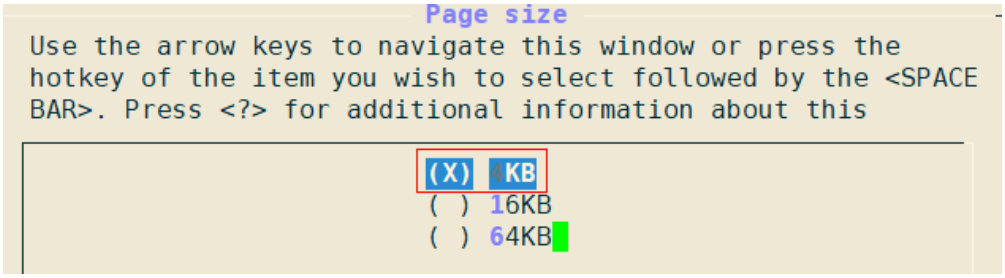
1. 选择“Kernel Features”。



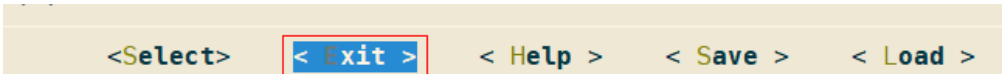
2. 按“Enter”键进入下一级菜单，选择“Page size（64KB）”。

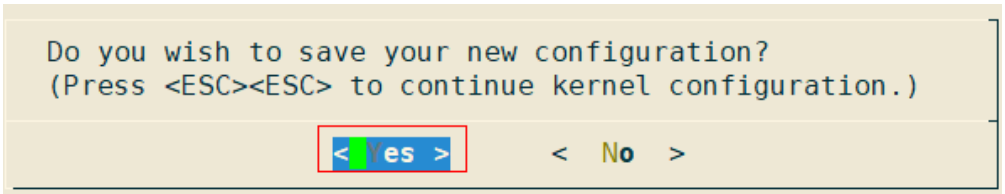


3. 按“Enter”键进入选择，按“Space”键选择4KB。



4. 按两次“exit”保存并退出。





步骤4 修改配置文件。

```
vi .config
```

手动在第一行添加“#aarm64”。


```
#aarm64
#
# Automatically generated file; DO NOT EDIT.
# Linux/arm64 4.14.0 Kernel Configuration
#
CONFIG_ARM64=y
CONFIG_64BIT=y
CONFIG_ARCH_PHYS_ADDR_T_64BIT=y
CONFIG_MMU=y
```

步骤5 确认配置文件。

1. 确认Bcache模块打开。

```
1926 # CONFIG_MD_MULTIPATH is not set
1927 CONFIG_MD_FAULTY=m
1928 CONFIG_BCACHE=m
1929 # CONFIG_BCACHE_DEBUG is not set
1930 # CONFIG_BCACHE_CLOSURES_DEBUG is not set
1931 CONFIG_BLK_DEV_DM_BUILTIN=y
```

2. 确认Pagesize为4k。

```
554 CONFIG_ARM64_4K_PAGES=y
555 # CONFIG_ARM64_16K_PAGES is not set
556 # CONFIG_ARM64_64K_PAGES is not set
557 # CONFIG_ARM64_VA_BITS_39 is not set
```

步骤6 覆盖默认配置文件。

```
cp .config ../kernel-alt-4.14.0-aarch64.config
```

输入y确认。

----结束

1.3.3 内核打补丁

说明

CentOS 7.6-aarch64 Linux内核版本4.14，该内核版本的Bcache模块在使用过程中，遇到了Bcache模块注册超时导致的机器启动失败的现象。我们提供了解决该问题的补丁，用户可以根据实际使用情况选择是否使用该补丁。

步骤1 获取补丁文件。

https://mirrors.huaweicloud.com/kunpeng/archive/kunpeng_solution/storage/Patch/

File Name ↓	File Size ↓	Date ↓
Parent directory/	-	-
bcache_patch.tar.gz	30.6 KiB	2020-Aug-06 09:18
ceph-14.2.8-common-rgw-add-openssl-engine-suppo..>	8.2 KiB	2020-Jun-11 09:58
ceph-14.2.8-zlib-compress.patch	37.6 KiB	2020-Aug-10 10:21

步骤2 拷贝补丁包至“/home”目录并解压。

```
cd /home
mkdir -p /home/bcache_patch
tar -zxvf bcache_patch.tar.gz -C /home/bcache_patch
```

步骤3 替换Bcache源码文件。

```
/bin/cp /home/bcache_patch/* /home/kernel-bcache/linux-4.14.0-115.10.1.el7a/drivers/md/bcache
```

----结束

1.3.4 生成内核 RPM 包

步骤1 内核源码重新打包。

```
cd /home/kernel-bcache
tar -cvf linux-4.14.0-115.10.1.el7a.tar linux-4.14.0-115.10.1.el7a
rm -f linux-4.14.0-115.10.1.el7a.tar.xz
xz -z linux-4.14.0-115.10.1.el7a.tar
```

步骤2 生成rpmbuild目录。

```
mkdir -p /home/rpmbuild
cd /home/rpmbuild
mkdir -p BUILD  RPMS  SOURCES  SPECS  SRPMS
```

步骤3 修改默认rpmbuild路径。

```
vi /root/.rpmmacros
```

增加如下内容：

```
%_topdir /home/rpmbuild
```

步骤4 拷贝文件至rpmbuild目录。

```
cp -r /home/kernel-bcache/* SOURCES/
cp /home/kernel-bcache/kernel-alt.spec SPECS/
rm -rf SOURCES/linux-4.14.0-115.10.1.el7a
```

步骤5 更新内核补丁。

```
rpmbuild -bp /home/rpmbuild/SPECS/kernel-alt.spec
```

步骤6 编译生成内核RPM包。

```
rpmbuild -bb /home/rpmbuild/SPECS/kernel-alt.spec --with baseonly --without debug --without debuginfo
Checking for unpackaged file(s): /usr/lib/rpm/check-files /home/rpmbuild/BUILDROOT/kernel-alt-4.14.0-115.10.1.el7a.aarch64
Wrote: /home/rpmbuild/RPMS/aarch64/kernel-4.14.0-115.10.1.el7a.aarch64.rpm
Wrote: /home/rpmbuild/RPMS/aarch64/kernel-headers-4.14.0-115.10.1.el7a.aarch64.rpm
Wrote: /home/rpmbuild/RPMS/aarch64/perf-4.14.0-115.10.1.el7a.aarch64.rpm
Wrote: /home/rpmbuild/RPMS/aarch64/python-perf-4.14.0-115.10.1.el7a.aarch64.rpm
Wrote: /home/rpmbuild/RPMS/aarch64/kernel-tools-4.14.0-115.10.1.el7a.aarch64.rpm
Wrote: /home/rpmbuild/RPMS/aarch64/kernel-tools-libs-4.14.0-115.10.1.el7a.aarch64.rpm
Wrote: /home/rpmbuild/RPMS/aarch64/kernel-tools-libs-devel-4.14.0-115.10.1.el7a.aarch64.rpm
Wrote: /home/rpmbuild/RPMS/aarch64/kernel-devel-4.14.0-115.10.1.el7a.aarch64.rpm
Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.4Lfz8M
+ umask 022
+ cd /home/rpmbuild/BUILD
+ cd kernel-alt-4.14.0-115.10.1.el7a
+ rm -rf /home/rpmbuild/BUILDROOT/kernel-alt-4.14.0-115.10.1.el7a.aarch64
+ exit 0
```

----结束

1.4 安装 Bcache 模块并验证

提供RPM包安装与ISO安装两种安装方式，可选择两种中的任意一种方式进行安装。

RPM 包安装

说明

- 采用内核RPM包安装需要提前在服务器安装对应的CentOS 7.6。
- 本章使用的RPM包为1.3.4 生成内核RPM包章节中生成的rpm包。

步骤1 将1.3.4 生成内核RPM包中生成的内核RPM包上传至“/home/kernel-rpm”目录。

```
mkdir -p /home/kernel-rpm
```

步骤2 安装内核RPM包。

```
cd /home/kernel-rpm
yum -y install kernel-4.14.0-115.10.1.el7a.aarch64.rpm kernel-devel-4.14.0-115.10.1.el7a.aarch64.rpm kernel-headers-4.14.0-115.10.1.el7a.aarch64.rpm
```

步骤3 查看默认内核启动项。

```
grub2-editenv list
```

```
[root@localhost centos-kernel-bcache-rpm]# grub2-editenv list
saved_entry=CentOS Linux (4.14.0-115.el7.0.1.aarch64) 7 (AltArch)
```

```
[root@localhost aarch64]# grub2-editenv list
saved_entry=openEuler (4.19.90-2003.4.0.0036.aarch64) 20.03 (LTS)
boot_success=0
```

查看其中的默认内核版本号。

- 如果默认版本是新安装的内核版本，则跳过步骤4。
- 如果默认版本不是新安装的内核版本，则执行步骤4。

步骤4 修改默认内核启动项。

```
cat /etc/grub2-efi.cfg  | grep CentOS
grub2-set-default "CentOS Linux (4.14.0-115.10.1.el7a.aarch64) 7 (AltArch)"
```

其中“CentOS Linux (4.14.0-115.10.1.el7a.aarch64) 7 (AltArch)”为新装内核版本。

步骤5 重启系统。

```
reboot
```

步骤6 验证Bcache模块。

```
modinfo bcache
```

```
[root@ceph1 ~]# modinfo bcache
filename:       /lib/modules/4.14.0-115.el7.0.1.aarch64/kernel/drivers/md/bcache/bcache.ko.xz
license:        GPL
author:         Kent Overstreet <koverstreet@google.com>
author:         Kent Overstreet <kent.overstreet@gmail.com>
license:        GPL
rhelversion:    7.6
srcversion:     F3FC8A01406453C1CAFE7D2
depends:
intree:         Y
name:           bcache
vermagic:       4.14.0-115.el7.0.1.aarch64 SMP mod_unload modversions aarch64
```

步骤7 验证pagesize。

```
getconf PAGESIZE
```

```
[root@ceph1 ~]# getconf PAGESIZE
4096
```

----结束

ISO 镜像安装

镜像安装首先需要生成已经修改内核以及添加bcache-tools的镜像。

步骤1 下载原生镜像文件并挂载至本地。

http://vault.centos.org/altarch/7.6.1810/isos/aarch64/

Parent Directory		-
0_README.txt	2018-12-03 15:23	1.7K
CentOS-7-aarch64-Everything-1810.iso	2018-11-27 11:14	7.5G
CentOS-7-aarch64-Minimal-1810.iso	2018-11-27 13:10	768M
CentOS-7-aarch64-NetInstall-1810.iso	2018-11-27 08:52	469M
sha256sum.txt	2018-11-27 16:30	306

```
mount CentOS-7-aarch64-Everything-7.6.iso /mnt/
```

步骤2 拷贝原生镜像所有文件至新目录。

```
mkdir -p /home/centos7.6-iso
cd /home/centos7.6-iso
cp -r /mnt/* ./
cp /mnt/.discinfo /mnt/.treeinfo ./
```

步骤3 使用**1.3.4 生成内核RPM包**中生成的RPM包替换原生内核RPM包。

```
cp /home/rpmbuild/RPMS/aarch64/* ./Packages
```

注意

生成的内核RPM包扩展名与镜像原生RPM包名不一样，需要先手动删除原RPM包。可以根据“/root/rpmbuild/RPMS/aarch64”下的RPM包名，使用命令**rm -f <包名>**删除对应的RPM包。

举例：

/home/centos7.6-iso/Packages/kernel-4.14.0-115.**el7**.0.1.aarch64.rpm

/home/centos7.6-iso/Packages/kernel-4.14.0-115.**el7a**.0.1.aarch64.rpm

则需要手动删除/home/centos7.6-iso/Packages/kernel-4.14.0-115.**el7a**.0.1.aarch64.rpm

步骤4 重新生成repodata。

```
createrepo -g repodata/aced7d22b338fdf7c0a71ffcf32614e058f4422c42476d1f4b9e9364d567702f-c7-x86_64-comps.xml ./
```

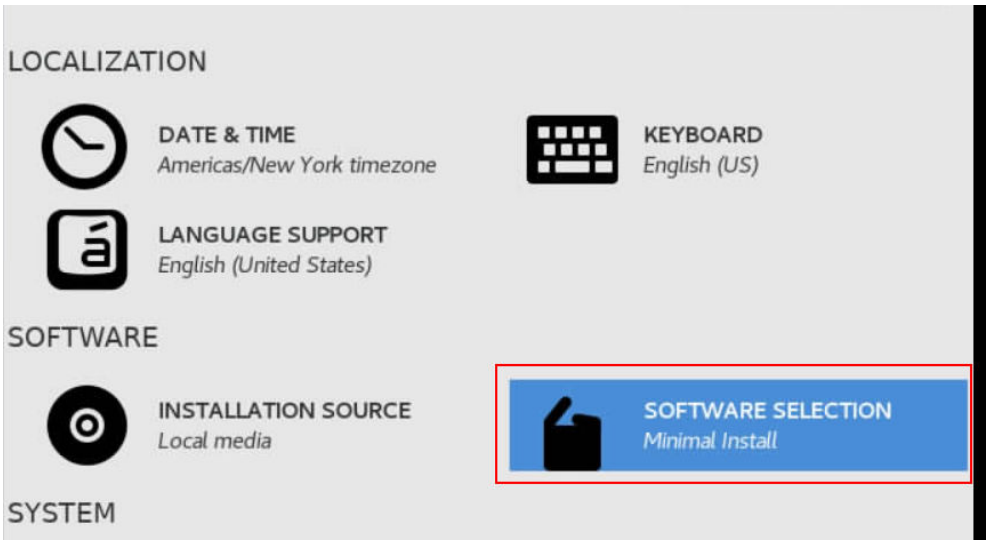
步骤5 生成ISO。

```
genisoimage -e images/efiboot.img -no-emul-boot -T -J -R -c boot.catalog -hide boot.catalog -V "CentOS 7 aarch64" -o /home/centos7.6-bcache.iso .
```

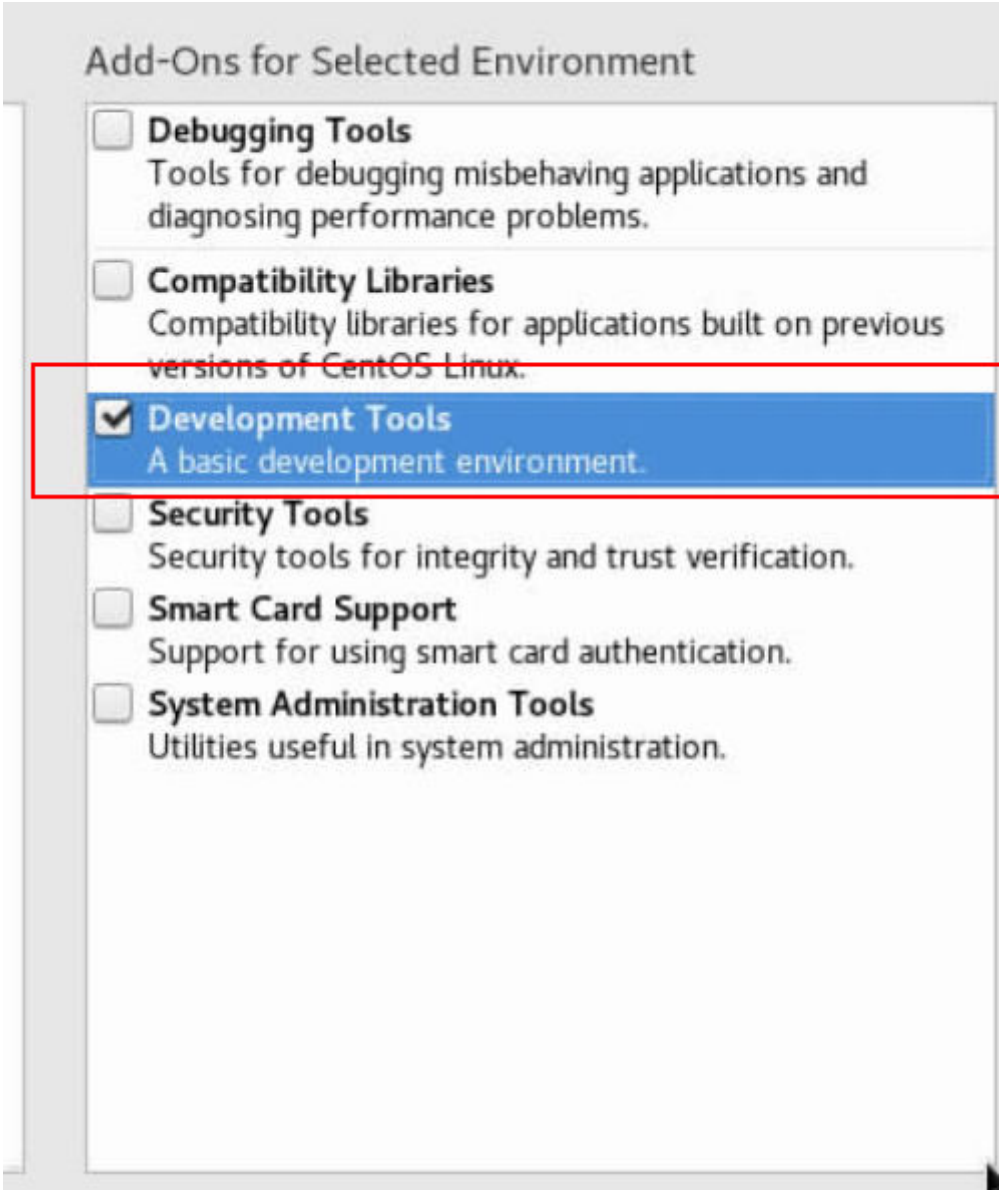
在“/home/”目录下生成镜像centos7.6-bcache.iso

步骤6 安装系统。

1. 在安装系统时选择SOFTWARE SELECTION。



2. 在选择需要的Base Environment后，在右侧勾选Development Tools。



3. 勾选后继续安装流程。

步骤7 验证PAGESIZE。

```
getconf PAGESIZE
```

```
[root@ceph1 ~]# getconf PAGESIZE
4096
```

步骤8 验证Bcache模块。

```
modinfo bcache
```

```
[root@ceph1 ~]# modinfo bcache
filename:       /lib/modules/4.14.0-115.el7.0.1.aarch64/kernel/drivers/md/bcache/bcache.ko.xz
license:        GPL
author:         Kent Overstreet <koverstreet@google.com>
author:         Kent Overstreet <kent.overstreet@gmail.com>
license:        GPL
rhelversion:    7.6
srcversion:     F3FC8A01406453C1CAFE7D2
depends:
intree:         Y
name:           bcache
vermagic:       4.14.0-115.el7.0.1.aarch64 SMP mod_unload modversions aarch64
```

----结束

1.5 Bcache 使用

1.5.1 安装 Bcache-tools 工具

Bcache通过**make-bcache**命令完成bcache的创建，需要安装相关的工具包bcache-tools。

下载链接：**https://github.com/g2p/bcache-tools/releases**。

版本：v1.0.8

操作步骤

步骤1 通过传输工具将下载的bcache-tools-1.0.8.tar.gz包上传到“/home”目录下。

步骤2 解压缩。

```
cd /home/
tar -zxvf bcache-tools-1.0.8.tar.gz
cd /home/bcache-tools-1.0.8
```

步骤3 安装依赖。

```
yum install libblkid-devel -y
```

步骤4 安装。

```
make
make install
```

步骤5 执行**make-bcache**命令，如下图所示。

```
[root@ceph1 bcache-tools-1.0.8]# make-bcache
Please supply a device
Usage: make-bcache [options] device
        -C, --cache             Format a cache device
        -B, --bdev              Format a backing device
        -b, --bucket            bucket size
        -w, --block             block size (hard sector size of SSD, often 2k)
        -o, --data-offset       data offset in sectors
            --cset-uuid         UUID for the cache set
            --writeback         enable writeback
            --discard           enable discards
            --cache_replacement_policy=(lru|fifo)
        -h, --help              display this help and exit
```

----结束

1.5.2 Bcache 基本操作

- 创建Bcache后端和缓存磁盘。

  ```
  make-bcache  -B /dev/sdx1 -C /dev/sdx2
  ```

  说明

  -B：指定后端磁盘设备，即数据盘。

  -C：指定缓存设备，用于加速数据盘。

  举例：后端磁盘为“sdb”，缓存设备为“sdc”。

  ```
  make-bcache  -B /dev/sdb -C /dev/sdc
  ```

  ```
  [root@ceph1 home]# lsblk
  NAME            MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
  sdd               8:48   0   1.1T  0 disk
  sdb               8:16   0   1.1T  0 disk
  └─bcache0       253:0    0   1.1T  0 disk
  sde               8:64   0 894.3G  0 disk
  sdc               8:32   0   1.1T  0 disk
  └─bcache0       253:0    0   1.1T  0 disk
  sda               8:0    0   1.1T  0 disk
  ├─sda2            8:2    0     1G  0 part /boot
  ├─sda3            8:3    0   1.1T  0 part
  │ ├─centos-swap 252:1    0     4G  0 lvm
  │ ├─centos-home 252:2    0     1T  0 lvm  /home
  │ └─centos-root 252:0    0    50G  0 lvm  /
  └─sda1            8:1    0   200M  0 part /boot/efi
  ```

- 删除缓存盘。

  a. 查看缓存盘的cset-uuid。

  ```
  bcache-super-show /dev/sd*
  ```

  举例：缓存设备为“sdc”。

  ```
  bcache-super-show /dev/sdc
  ```

```
[root@ceph1 home]# bcache-super-show /dev/sdc
sb.magic                ok
sb.first_sector         8 [match]
sb.csum                 6CCC4B079543F8C5 [match]
sb.version              3 [cache device]

dev.label               (empty)
dev.uuid                367fdbd3-b163-45c4-a2b1-40f9ec4ab3ea
dev.sectors_per_block   1
dev.sectors_per_bucket  1024
dev.cache.first_sector  1024
dev.cache.cache_sectors 2344224768
dev.cache.total_sectors 2344225792
dev.cache.ordered       yes
dev.cache.discard       no
dev.cache.pos           0
dev.cache.replacement   0 [lru]

cset.uuid               5f50eddf-69d8-45e3-9b67-7386ffdaceb7
```

b. 删除缓存操作。

```
echo cset-uuid  > /sys/block/bcache<n>/bcache/detach
```

举例：cset-uuid为“5f50eddf-69d8-45e3-9b67-7386ffdaceb7”。

```
echo 5f50eddf-69d8-45e3-9b67-7386ffdaceb7 > /sys/block/bcache0/bcache/detach
```

此时sdc与bcache0解除绑定。

说明

若需要恢复缓存，可通过以下命令重新绑定缓存。

```
echo cset-uuid  > /sys/block/bcache<n>/bcache/attach
```

- 注销缓存盘。

```
echo 1  > /sys/fs/bcache/<cset-uuid>/unregister
```

举例：cset-uuid为“5f50eddf-69d8-45e3-9b67-7386ffdaceb7”。

```
echo 1  > /sys/fs/bcache/5f50eddf-69d8-45e3-9b67-7386ffdaceb7/unregister
```

- 停用缓存盘。

```
echo 1  > /sys/fs/bcache/<cset-uuid>/stop
```

举例：cset-uuid为“5f50eddf-69d8-45e3-9b67-7386ffdaceb7”。

```
echo 1  > /sys/fs/bcache/5f50eddf-69d8-45e3-9b67-7386ffdaceb7/stop
```

- 停用后端设备。

```
echo 1 > /sys/block/bcache<n>/bcache/stop
```

举例：n为“0”。

```
echo 1 > /sys/block/bcache0/bcache/stop
```

此时 sdb，sdc均与bcache0解除绑定关系。

```
[root@ceph1 home]# lsblk
NAME            MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sdd               8:48   0   1.1T  0 disk
sdb               8:16   0   1.1T  0 disk
sde               8:64   0 894.3G  0 disk
sdc               8:32   0   1.1T  0 disk
sda               8:0    0   1.1T  0 disk
├─sda2            8:2    0     1G  0 part /boot
├─sda3            8:3    0   1.1T  0 part
│ ├─centos-swap 252:1    0     4G  0 lvm
│ ├─centos-home 252:2    0     1T  0 lvm  /home
│ └─centos-root 252:0    0    50G  0 lvm  /
└─sda1            8:1    0   200M  0 part /boot/efi
```

- 卸载bcache模块。
  ```
  rmmod bcache
  ```

  须知

  卸载后Bcache将无法使用，请谨慎执行此操作。

2 Bcache 用户指南（openEuler 20.03）

2.1 介绍

Bcache是Linux内核块层cache，它使用SSD来作为HDD硬盘的cache，从而起到加速作用。Bcache内核模块仅在Linux 3.10及以上版本支持，因此使用Bcache，需要将内核升级到3.10及以上版本，并在内核配置项中打开Bcache模块。

为了方便Bcache的使用，本文提供指导流程，可分别制作：

- Bcache模块内核RPM包
- Bcache-tools RPM包
- 包含Bcache内核RPM包、Bcache-tools RPM包的iso镜像

更多详情见Bcache主页和使用手册：

https://bcache.evilpiepirate.org/

https://evilpiepirate.org/git/linux-bcache.git/tree/Documentation/bcache.txt

2.2 环境准备

环境要求

环境要求如下表所示。

服务器名称	TaiShan 200服务器（型号2280）
OS	openEuler 20.03

说明

- 由于制作安装包过程中需要在线安装依赖包，请确保服务器可以接入互联网或已配置本地源。
- 本文档提供基于openEuler 20.03的4.19内核版本的指导流程，其他版本可参考本文档执行。
- 请确保服务器安装OS版本与要修改OS版本一致。

2.3 编译内核

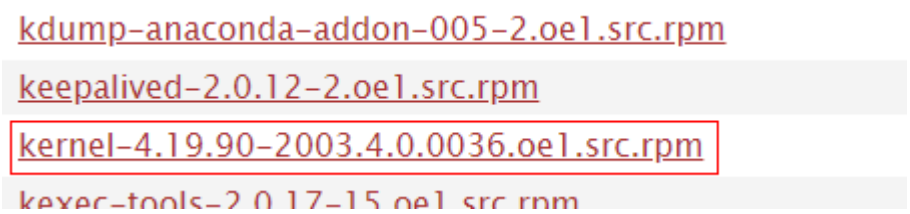
2.3.1 获取源码

步骤1 创建目录并进入该目录。

```
mkdir -p /home/kernel-bcache
cd /home/kernel-bcache
```

步骤2 下载内核源码包至“/home/kernel-bcache”目录。

下载路径：**https://repo.openeuler.org/openEuler-20.03-LTS/source/Packages/**



步骤3 解压源码包。

```
rpm2cpio kernel-4.19.90-2003.4.0.0036.oe1.src.rpm | cpio -divm
tar -zxvf kernel.tar.gz
```

步骤4 先将OS对应的everything镜像源文件上传到服务器。

通过SFTP工具将“openEuler-***-everything-aarch64-dvd.iso”上传到服务器上“/root”目录下。

步骤5 创建一个本地文件夹用于挂载本地镜像。

```
mkdir -p /iso
```

说明

此处以在根目录下创建iso文件夹为例。

步骤6 将iso文件挂载到本地文件夹。

```
mount /root/openEuler-***-everything-aarch64-dvd.iso /iso
```

步骤7 创建镜像yum源。

```
vi /etc/yum.repos.d/openEuler.repo
```

在文件中加入以下内容：

```
[Base]
name=Base
baseurl=file:///iso
enabled=1
```

```
gpgcheck=0
priority=1
```

步骤8 配置yum源。

```
vi /etc/yum.repos.d/openEuler.repo
```

在文件中加入以下内容：

```
[openEuler-source]
name=openEuler-source
baseurl=http://repo.openeuler.org/openEuler-20.03-LTS/source/
enabled=1
gpgcheck=1
gpgkey=http://repo.openeuler.org/openEuler-20.03-LTS/source/RPM-GPG-KEY-openEuler

[openEuler-os]
name=openEuler-os
baseurl=http://repo.openeuler.org/openEuler-20.03-LTS/OS/aarch64/
enabled=0
gpgcheck=1
gpgkey=http://repo.openeuler.org/openEuler-20.03-LTS/OS/aarch64/RPM-GPG-KEY-openEuler

[openEuler-everything]
name=openEuler-everything
baseurl=http://repo.openeuler.org/openEuler-20.03-LTS/everything/aarch64/
enabled=0
gpgcheck=1
gpgkey=http://repo.openeuler.org/openEuler-20.03-LTS/everything/aarch64/RPM-GPG-KEY-openEuler

[openEuler-EPOL]
name=openEuler-epol
baseurl=http://repo.openeuler.org/openEuler-20.03-LTS/EPOL/aarch64/
enabled=1
gpgcheck=0

[openEuler-update]
name=openEuler-update
baseurl=http://repo.openeuler.org/openEuler-20.03-LTS/update/aarch64/
enabled=1
gpgcheck=0
```

步骤9 安装依赖。

```
dnf -y install ncurses-devel bison m4 flex rpm-build rpmdevtools asciidoc audit-libs-devel binutils-devel elfutils-devel elfutils-libelf-devel gtk2-devel java-1.8.0-openjdk-devel xz-devel libbabeltrace-devel libunwind-devel newt-devel numactl-devel openssl-devel pciutils-devel perl-generators python3-devel python3-docutils xmlto zlib-devel mkeuleros createrepo genisoimage
```

----结束

2.3.2 修改内核配置

步骤1 获取内核配置文件。

```
cd /home/kernel-bcache/kernel
cp arch/arm64/configs/openeuler_defconfig .config
```

步骤2 配置Bcache模块。

```
make menuconfig
```

1. 选择“Device Drivers ”。

```
    General setup  --->
[*] Enable loadable module support  --->
-*- Enable the block layer  --->
    Platform selection  --->
    Bus support  --->
    Kernel Features  --->
    Boot options  --->
    Userspace binary formats  --->
    Power management options  --->
    CPU Power Management  --->
[*] Networking support  --->
    Device Drivers  --->
```

2. 按“Enter”键进入下一级菜单，选择“Multiple device driver support (RAID and LVM)”。

```
[*] Block devices  --->
<M> NVM Express block device
<M> NVM Express over Fabrics RDMA host driver
< > NVM Express over Fabrics FC host driver
<M> NVMe Target support
<M>   NVMe loopback device support
<M>   NVMe over Fabrics RDMA target support
< >   NVMe over Fabrics FC target driver
    Misc devices  --->
    SCSI device support  --->
<*> Serial ATA and Parallel ATA drivers (libata)  --->
[*] Multiple devices driver support (RAID and LVM)  --->
<M> Generic Target Core Mod (TCM) and ConfigFS Infrastructure  --->
```

3. 按“Enter”键进入下一级菜单，选中“Block device as cache”，键盘输入“M”选中该配置。

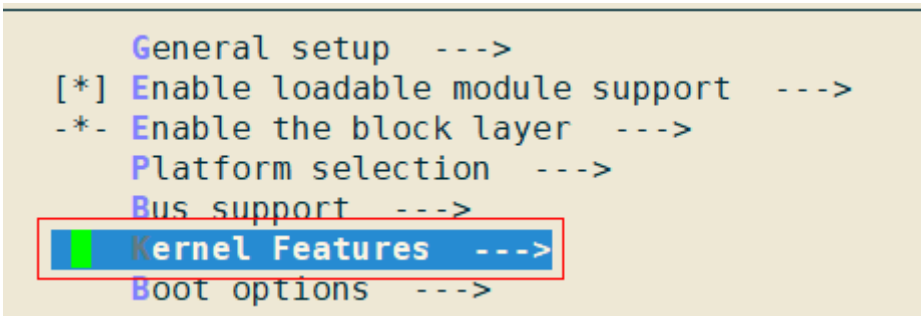
```
--- Multiple devices driver support (RAID and LVM)
{*}   RAID support
[*]     Autodetect RAID arrays during kernel boot
<M>     Linear (append) mode
{M}     RAID-0 (striping) mode
{M}     RAID-1 (mirroring) mode
{M}     RAID-10 (mirrored striping) mode
{M}     RAID-4/RAID-5/RAID-6 mode
< >     Multipath I/O support
<M>     Faulty test module for MD
<M>     Block device as cache
[ ]     Bcache debugging
```

4. 按两次“exit”返回至第一层。

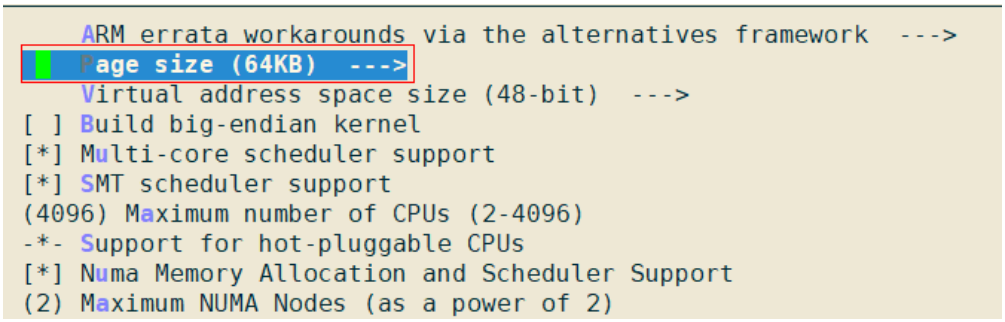
```
    <Select>   < Exit >   < Help >   < Save >   < Load >
```

步骤3 修改内核PAGESIZE大小为4K。

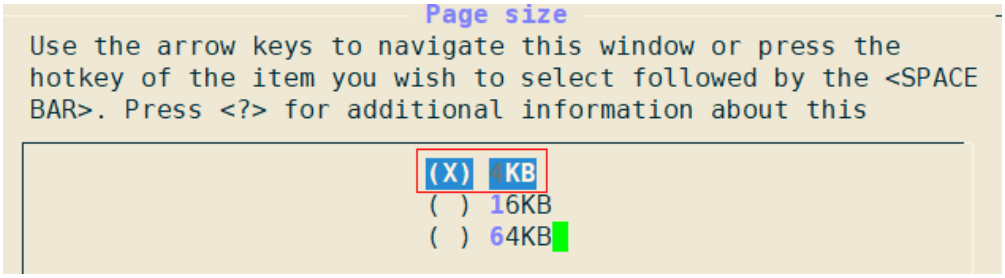
1. 选择“Kernel Features”。



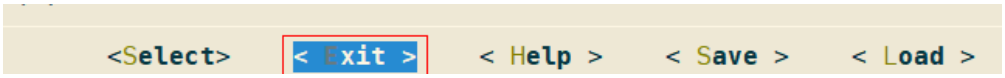
2. 按“Enter”键进入下一级菜单，选择“Page size（64KB）”。

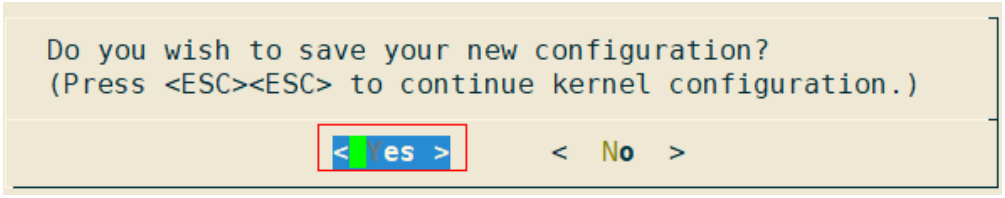


3. 按“Enter”键进入选择，按“Space”键选择4KB。



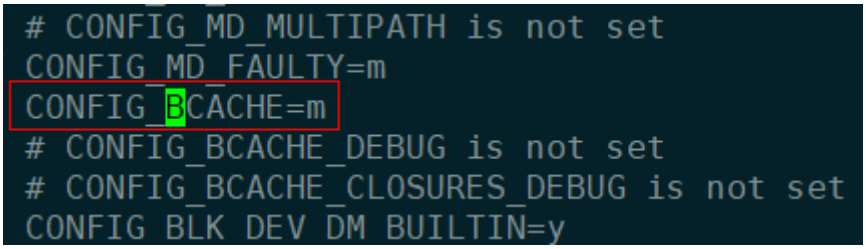
4. 按两次“exit”保存并退出。



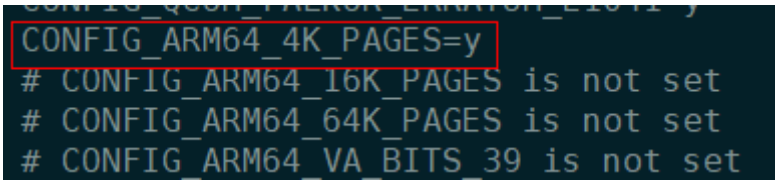


步骤4 确认配置文件。

1. 确认Bcache模块打开。



2. 确认Pagesize为4K。



步骤5 覆盖默认配置文件。

```
cp .config arch/arm64/configs/openeuler_defconfig
```

输入y确认

----结束

2.3.3 生成内核 RPM 包

步骤1 内核源码重新打包。

```
cd /home/kernel-bcache
tar -zcvf kernel.tar.gz kernel
```

步骤2 生成rpmbuild目录。

```
mkdir -p /home/rpmbuild
cd /home/rpmbuild
mkdir -p BUILD  RPMS  SOURCES  SPECS  SRPMS
```

步骤3 修改默认rpmbuild路径。

```
vi /root/.rpmmacros
```

增加如下内容：

```
%_topdir /home/rpmbuild
```

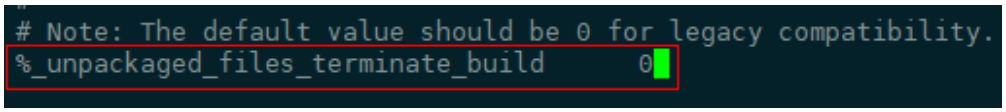
步骤4 拷贝文件至rpmbuild目录。

```
cp -r /home/kernel-bcache/* SOURCES/
cp /home/kernel-bcache/kernel.spec SPECS/
rm -rf SOURCES/kernel
```

步骤5 编译生成内核RPM包。

```
vi /usr/lib/rpm/macros
```

修改“%_unpackaged_files_terminate_build 1”为“%_unpackaged_files_terminate_build 0”。




```
rpmbuild -bb /home/rpmbuild/SPECS/kernel.spec --with baseonly --without debug --without debuginfo
```

----结束

2.4 安装 Bcache 模块并验证

提供RPM包安装与ISO安装两种安装方式，可选择两种中的任意一种方式进行安装。

RPM 包安装

说明

- 采用内核RPM包安装需要提前在服务器安装对应的openEuler 20.03。
- 本章使用的RPM包为2.3.3 生成内核RPM包章节中生成的rpm包。

步骤1 将2.3.3 生成内核RPM包中生成的内核RPM包上传至“/home/kernel-rpm”目录。

```
mkdir -p /home/kernel-rpm
cp /home/rpmbuild/RPMS/aarch64/* /home/kernel-rpm
```

步骤2 安装内核RPM包。

```
cd /home/kernel-rpm
dnf -y install kernel-4.19.90-2003.4.0.0036.aarch64.rpm kernel-
devel-4.19.90-2003.4.0.0036.aarch64.rpm
```

步骤3 安装依赖。

```
yum install libblkid
```

步骤4 查看默认内核启动项。

```
grub2-editenv list
```

```
[root@localhost centos-kernel-bcache-rpm]# grub2-editenv list
saved_entry=CentOS Linux (4.14.0-115.el7.0.1.aarch64) 7 (AltArch)
```

```
[root@localhost aarch64]# grub2-editenv list
saved_entry=openEuler (4.19.90-2003.4.0.0036.aarch64) 20.03 (LTS)
boot_success=0
```

查看其中的默认内核版本号。

- 如果默认版本是新安装的内核版本，则跳过步骤5。
- 如果默认版本不是新安装的内核版本，则执行步骤5。

步骤5 修改默认内核启动项。

```
cat /etc/grub2-efi.cfg  | grep openEuler
grub2-set-default "openEuler (4.19.90-2003.4.0.0036.aarch64) 20.03 (LTS)"
```

```
[root@localhost aarch64]# cat /etc/grub2-efi.cfg  | grep openEuler
menuentry 'openEuler (4.19.90-2003.4.0.0036.aarch64) 20.03 (LTS)' --class openeuler --class gnu-linux --class gnu --c
f-0116119c5f0d' {
menuentry 'openEuler (4.19.90-2003.4.0.0036.oe1.aarch64) 20.03 (LTS)' --class openeuler --class gnu-linux --class gnu
-b5bf-0116119c5f0d' {
menuentry 'openEuler (0-rescue-1561167f623844618bdeb1d06dc7aad2) 20.03 (LTS)' --class openeuler --class gnu-linux --c
100d94-c542-431d-b5bf-0116119c5f0d' {
if [ "x$default" = 'openEuler (4.19.90-2003.4.0.0036.oe1.aarch64) 20.03 (LTS)' ]; then default='Advanced options for
```

其中“openEuler (4.19.90-2003.4.0.0036.aarch64) 20.03 (LTS)”为新安装内核版本。

步骤6 重启系统。

```
reboot
```

步骤7 验证Bcache模块。

```
modinfo bcache
```

```
[root@ceph1 ~]# modinfo bcache
filename:       /lib/modules/4.14.0-115.el7.0.1.aarch64/kernel/drivers/md/bcache/bcache.ko.xz
license:        GPL
author:         Kent Overstreet <koverstreet@google.com>
author:         Kent Overstreet <kent.overstreet@gmail.com>
license:        GPL
rhelversion:    7.6
srcversion:     F3FC8A01406453C1CAFE7D2
depends:
intree:         Y
name:           bcache
vermagic:       4.14.0-115.el7.0.1.aarch64 SMP mod_unload modversions aarch64
```

步骤8 验证pagesize。

```
getconf PAGESIZE
```

```
[root@ceph1 ~]# getconf PAGESIZE
4096
```

----结束

ISO 镜像安装

步骤1 下载原生镜像文件并挂载至本地。

https://repo.openeuler.org/openEuler-20.03-LTS/ISO/aarch64/

Index of /openEuler-20.03-LTS/ISO/aarch64/

File Name ↓	File Size ↓
Parent directory/	-
openEuler-20.03-LTS-aarch64-dvd.iso	4.3 GiB
openEuler-20.03-LTS-aarch64-dvd.iso.sha256sum	102 B
openEuler-20.03-LTS-debuginfo-aarch64-dvd.iso	10.9 GiB
openEuler-20.03-LTS-debuginfo-aarch64-dvd.iso.s..>	112 B
openEuler-20.03-LTS-everything-aarch64-dvd.iso	10.9 GiB
openEuler-20.03-LTS-everything-aarch64-dvd.iso...>	113 B

```
mount openEuler-20.03-LTS-everything-aarch64-dvd.iso /mnt/
```

步骤2 拷贝原生镜像所有文件至新目录。

```
mkdir -p /home/openEuler20.03-iso
cd /home/openEuler20.03-iso
cp -r /mnt/* ./
cp /mnt/.discinfo /mnt/.treeinfo ./
```

步骤3 使用2.3.3 生成内核RPM包中生成的rpm包替换原生内核rpm包。

```
cp /home/rpmbuild/RPMS/aarch64/* ./Packages
```

注意

生成的内核RPM包扩展名与镜像原生RPM包名不一样，需要先手动删除原RPM包。可以根据“/root/rpmbuild/RPMS/aarch64”下的RPM包名，使用命令**rm -f <包名>**删除对应的原有RPM包。

举例：

（新）/home/openEuler20.03-iso/Packages/
kernel-4.19.90-2003.4.0.0036.aarch64.rpm

（旧）/home/openEuler20.03-iso/Packages/
kernel-4.19.90-2003.4.0.0036.**oe1**.aarch64.rpm

则需要手动删除/home/openEuler20.03-iso/Packages/
kernel-4.19.90-2003.4.0.0036.**oe1**.aarch64.rpm

通过**ls|grep kernel** 查看所有冲突内核RPM包

步骤4 重新生成repodata。

```
createrepo -g repodata/normal.xml ./
```

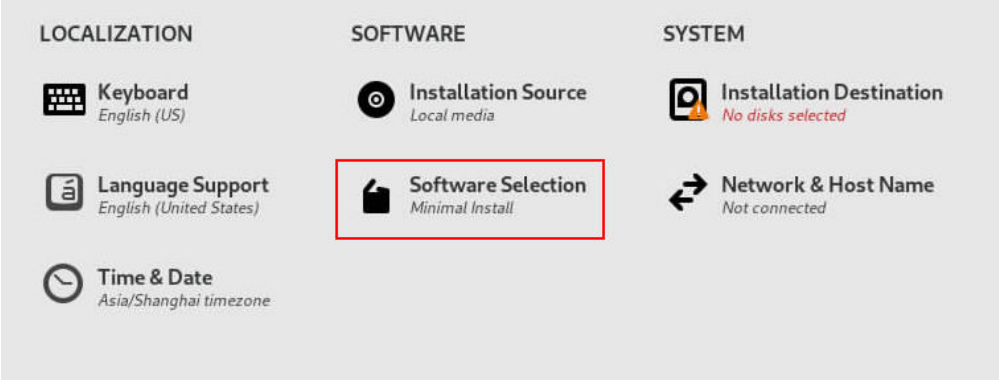
步骤5 生成ISO。

```
cd /opt/mkeuleros/
sh mkeuleros.sh -f config/aarch64/standard.conf -n openEuler -v 20.03-LTS -s SP1 -a aarch64 -r file:///home/openEuler20.03-iso/
```

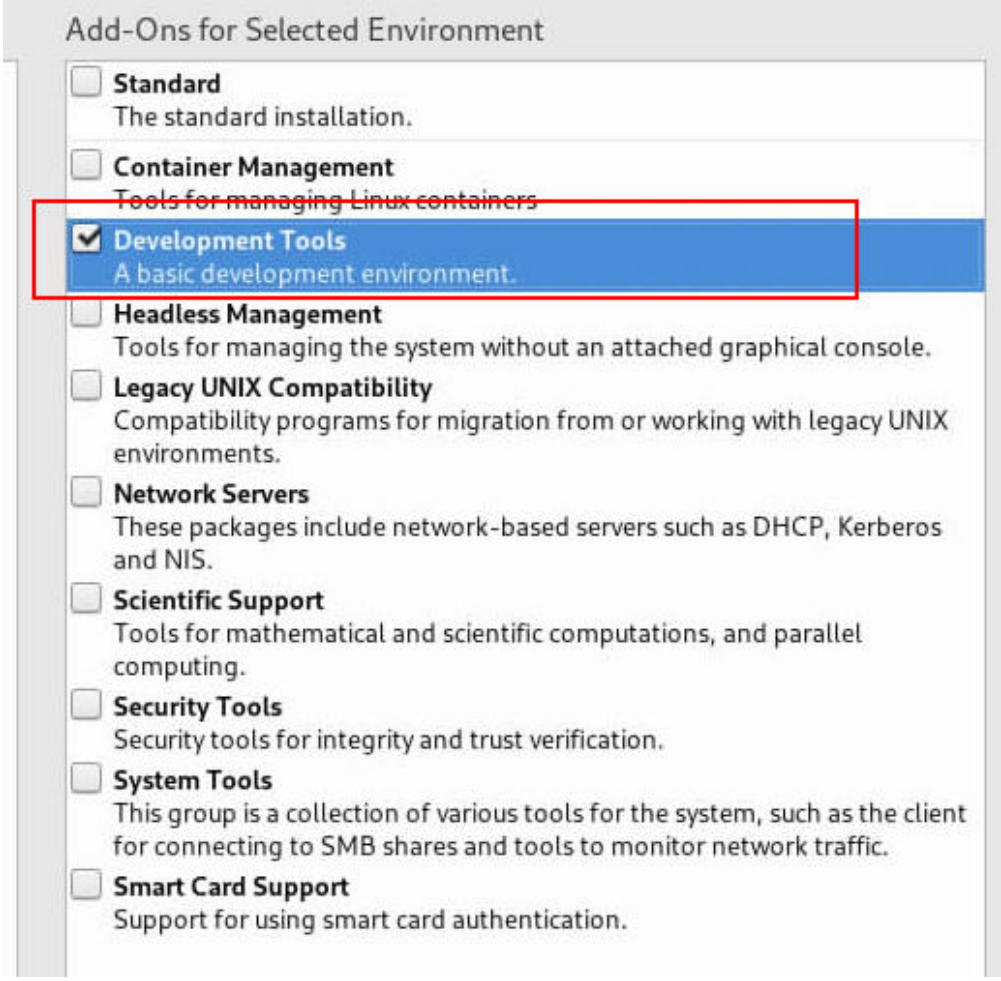
在“/result/”目录下生成镜像openEuler-20.03-LTS-aarch64-dvd.iso。

步骤6 使用步骤5生成的镜像安装系统。

1. 在安装系统时选择SOFTWARE SELECTION。



2. 在选择需要的Base Environment后，在右侧勾选Development Tools。



3. 勾选后继续安装流程。

步骤7 验证PAGESIZE。

getconf PAGESIZE

```
[root@ceph1 ~]# getconf PAGESIZE
4096
```

步骤8 验证Bcache模块。

modinfo bcache

```
[root@ceph1 ~]# modinfo bcache
filename:       /lib/modules/4.14.0-115.el7.0.1.aarch64/kernel/drivers/md/bcache/bcache.ko.xz
license:        GPL
author:         Kent Overstreet <koverstreet@google.com>
author:         Kent Overstreet <kent.overstreet@gmail.com>
license:        GPL
rhelversion:    7.6
srcversion:     F3FC8A01406453C1CAFE7D2
depends:
intree:         Y
name:           bcache
vermagic:       4.14.0-115.el7.0.1.aarch64 SMP mod_unload modversions aarch64
```

步骤9 （可选）验证1822驱动。

modinfo hinic

```
[root@ceph1 ~]# modinfo hinic
filename:       /lib/modules/4.14.0-115.el7.0.1.aarch64/kernel/drivers/net/ethernet/huawei/hinic/hinic.ko.xz
license:        GPL
version:        1.8.3.3
description:    Huawei(R) Intelligent Network Interface Card Driver
author:         Huawei Technologies CO., Ltd
rhelversion:    7.6
srcversion:     E0527C7681D8EFD0C23F6EA
alias:          pci:v000019E5d00000211sv*sd*bc*sc*i*
alias:          pci:v000019E5d00000210sv*sd*bc*sc*i*
alias:          pci:v000019E5d00000205sv*sd*bc*sc*i*
alias:          pci:v000019E5d0000020Dsv*sd*bc*sc*i*
alias:          pci:v000019E5d00000204sv*sd*bc*sc*i*
alias:          pci:v000019E5d00000208sv*sd*bc*sc*i*
alias:          pci:v000019E5d0000020Bsv*sd*bc*sc*i*
alias:          pci:v000019E5d0000379Esv*sd*bc*sc*i*
alias:          pci:v000019E5d0000375Esv*sd*bc*sc*i*
alias:          pci:v000019E5d00001822sv*sd*bc*sc*i*
depends:
intree:         Y
name:           hinic
vermagic:       4.14.0-115.el7.0.1.aarch64 SMP mod_unload modversions aarch64
parm:           disable_vf_load:Disable virtual functions probe or not - default is false (bool)
parm:           num_qps:Number of Queue Pairs (default unset) (ushort)
parm:           ovs_num_qps:Number of Queue Pairs in ovs mode (default=16) (ushort)
parm:           poll_weight:Number packets for NAPI budget (default=64) (uint)
parm:           qp_pending_limit:QP MSI-X Interrupt coalescing parameter pending_limit (default=2) (byte)
parm:           qp_coalesc_timer_cfg:QP MSI-X Interrupt coalescing parameter coalesc_timer_cfg (default=32) (byte)
parm:           set_max_wqe_num:Set lro max wqe number, valid range is 1 - 32, default is 4(arm) / 8(x86) (byte)
parm:           rx_buff:Set rx_buff size, buffer len must be 2^n. 2 - 16, default is 2KB (ushort)
parm:           set_lro_timer:Set lro timer in micro second, valid range is 1 - 1024, default is 16 (uint)
parm:           set_link_status_follow:Set link status follow port status. 0 - default, 1 - follow, 2 - separate, other - unset. (default unset) (byte)
parm:           lro_replenish_thld:Number wqe for lro replenish buffer (default=256) (uint)
parm:           l2nic_interrupt_switch:Control whether execute l2nic io interrupt switch or not, default is true (bool)
parm:           lro_en_status:lro enable status. 0 - disable, 1 - enable, other - unset. (default unset) (byte)
parm:           g_aeq_len:aeq depth, valid range is 64 - (512 * 1024) (uint)
parm:           g_ceq_len:ceq depth, valid range is 64 - (1024 * 1024) (uint)
parm:           g_num_ceqe_in_tasklet:The max number of ceqe can be processed in tasklet, default = 1024 (uint)
parm:           tx_pending_limit:TX CI coalescing parameter pending_limit (default=0) (byte)
parm:           tx_coalescing_time:TX CI coalescing parameter coalescing_time (default=0) (byte)
parm:           set_vf_link_state:Set vf link state, 0 represents link auto, 1 represents link always up, 2 represents link always down. - default is 0. (byte)
```

----结束

2.5 Bcache 使用

2.5.1 安装 Bcache-tools 工具

Bcache通过**make-bcache**命令完成bcache的创建，需要安装相关的工具包bcache-tools。

下载链接：。**https://git.kernel.org/pub/scm/linux/kernel/git/colyli/bcache-tools.git/tag/?h=bcache-tools-1.1**

版本：v1.1

操作步骤

步骤1 通过传输工具将下载的bcache-tools-1.1.tar.gz包上传到“/home”目录下。

步骤2 解压缩。

```
cd /home/
tar -zxvf bcache-tools-1.1.tar.gz
cd /home/bcache-tools-1.1
```

步骤3 安装依赖。

```
yum install libblkid-devel -y
```

步骤4 安装。

```
make
make install
```

步骤5 执行**make-bcache**命令，如下图所示。

```
[root@ceph1 bcache-tools-1.0.8]# make-bcache
Please supply a device
Usage: make-bcache [options] device
       -C, --cache             Format a cache device
       -B, --bdev              Format a backing device
       -b, --bucket            bucket size
       -w, --block             block size (hard sector size of SSD, often 2k)
       -o, --data-offset       data offset in sectors
           --cset-uuid         UUID for the cache set
           --writeback         enable writeback
           --discard           enable discards
           --cache_replacement_policy=(lru|fifo)
       -h, --help              display this help and exit
```

----结束

2.5.2 Bcache 基本操作

- 创建Bcache后端和缓存磁盘。

  ```
  make-bcache  -B /dev/sdx1 -C /dev/sdx2
  ```

  说明

  -B：指定后端磁盘设备，即数据盘。

  -C：指定缓存设备，用于加速数据盘。

  举例：后端磁盘为“sdb”，缓存设备为“sdc”。

  ```
  make-bcache  -B /dev/sdb -C /dev/sdc
  ```

  ```
  [root@ceph1 home]# lsblk
  NAME            MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
  sdd               8:48   0   1.1T  0 disk
  sdb               8:16   0   1.1T  0 disk
  └─bcache0       253:0    0   1.1T  0 disk
  sde               8:64   0 894.3G  0 disk
  sdc               8:32   0   1.1T  0 disk
  └─bcache0       253:0    0   1.1T  0 disk
  sda               8:0    0   1.1T  0 disk
  ├─sda2            8:2    0     1G  0 part /boot
  ├─sda3            8:3    0   1.1T  0 part
  │ ├─centos-swap 252:1    0     4G  0 lvm
  │ ├─centos-home 252:2    0     1T  0 lvm  /home
  │ └─centos-root 252:0    0    50G  0 lvm  /
  └─sda1            8:1    0   200M  0 part /boot/efi
  ```

- 删除缓存盘。

a. 查看缓存盘的cset-uuid。

```
bcache-super-show /dev/sd*
```

举例：缓存设备为“sdc”。

```
bcache-super-show /dev/sdc
```

```
[root@ceph1 home]# bcache-super-show /dev/sdc
sb.magic                ok
sb.first_sector         8 [match]
sb.csum                 6CCC4B079543F8C5 [match]
sb.version              3 [cache device]

dev.label               (empty)
dev.uuid                367fdbd3-b163-45c4-a2b1-40f9ec4ab3ea
dev.sectors_per_block   1
dev.sectors_per_bucket  1024
dev.cache.first_sector  1024
dev.cache.cache_sectors 2344224768
dev.cache.total_sectors 2344225792
dev.cache.ordered       yes
dev.cache.discard       no
dev.cache.pos           0
dev.cache.replacement   0 [lru]

cset.uuid               5f50eddf-69d8-45e3-9b67-7386ffdaceb7
[root@ceph1 home]#
```

b. 删除缓存操作。

```
echo cset-uuid  > /sys/block/bcache<n>/bcache/detach
```

举例：cset-uuid为“5f50eddf-69d8-45e3-9b67-7386ffdaceb7”。

```
echo 5f50eddf-69d8-45e3-9b67-7386ffdaceb7 > /sys/block/bcache0/bcache/detach
```

此时sdc与bcache0解除绑定。

说明

若需要恢复缓存，可通过以下命令重新绑定缓存。

```
echo cset-uuid  > /sys/block/bcache<n>/bcache/attach
```

- 注销缓存盘。

```
echo 1  > /sys/fs/bcache/<cset-uuid>/unregister
```

举例：cset-uuid为“5f50eddf-69d8-45e3-9b67-7386ffdaceb7”。

```
echo 1  > /sys/fs/bcache/5f50eddf-69d8-45e3-9b67-7386ffdaceb7/unregister
```

- 停用缓存盘。

```
echo 1  > /sys/fs/bcache/<cset-uuid>/stop
```

举例：cset-uuid为“5f50eddf-69d8-45e3-9b67-7386ffdaceb7”。

```
echo 1  > /sys/fs/bcache/5f50eddf-69d8-45e3-9b67-7386ffdaceb7/stop
```

- 停用后端设备。

```
echo 1 > /sys/block/bcache<n>/bcache/stop
```

举例：n为“0”。

```
echo 1 > /sys/block/bcache0/bcache/stop
```

此时 sdb，sdc均与bcache0解除绑定关系。

```
[root@ceph1 home]# lsblk
NAME            MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sdd               8:48   0   1.1T  0 disk
sdb               8:16   0   1.1T  0 disk
sde               8:64   0 894.3G  0 disk
sdc               8:32   0   1.1T  0 disk
sda               8:0    0   1.1T  0 disk
├─sda2            8:2    0     1G  0 part /boot
├─sda3            8:3    0   1.1T  0 part
│ ├─centos-swap 252:1    0     4G  0 lvm
│ ├─centos-home 252:2    0     1T  0 lvm  /home
│ └─centos-root 252:0    0    50G  0 lvm  /
└─sda1            8:1    0   200M  0 part /boot/efi
```

- 卸载bcache模块。

```
rmmod bcache
```

须知

卸载后Bcache将无法使用，请谨慎执行此操作。

3 IO 直通工具 用户指南

3.1 介绍

IO直通工具是针对Ceph均衡型场景下的一个流程优化工具，可以自动对Ceph集群进行性能优化。

3.2 兼容规格

硬件兼容列表

兼容项目	兼容性规格描述
CPU	华为鲲鹏920处理器
服务器型号	TaiShan 200服务器（型号2280）
	TaiShan 200服务器（型号5280）
磁盘控制器	直通背板
	HBA卡
	RAID卡（JBOD模式）： Avago 3408、Avago 3416、Avago 3508、PM8204、PM8222

软件兼容列表

兼容项目	兼容性规格描述
OS	CentOS Linux release 7.6.1810
	openEuler 20.03
Ceph	Ceph 12.2.x
	Ceph 14.2.x

3.3 适用场景

IO直通工具是针对均衡型场景的OSD存储IO流程进行优化，仅针对采用BlueStore存储引擎、Data与DB分区位于不同磁盘的OSD。

考虑到实际应用中Ceph集群中OSD的增加与修改操作，工具需要获取集群内的OSD信息，因此需要保证OSD部署路径使用官方默认路径“/var/lib/ceph/osd”，如果采用其他路径则不会进行优化。

说明

存储场景介绍如下：

- 均衡型：集群采用SSD盘和HDD盘混合部署OSD，使用SSD盘存储OSD的元数据（DB）、日志数据（WAL），使用HDD盘存储真实数据（Data）。
- 全闪存：集群采用全SSD盘部署OSD，OSD的Data、DB、WAL一起存储在SSD盘中。
- 冷存储：集群采用全HDD盘部署OSD，OSD的Data、DB、WAL一起存储在HDD盘中。
- Bcache：将SSD与HDD绑定为一个Bcache分区，SSD做为HDD的缓存，集群采用Bcache分区部署OSD，OSD的Data、DB、WAL一起存储在Bcache分区中。

目前IO直通工具只对均衡型场景有效。

3.4 使用指导

步骤1 获取OS对应RPM包和数字签名文件。

- 华为企业网网站下载
  - CentOS软件包：**ceph-boost-1.0.1-centos.aarch64.rpm**
  - openEuler软件包：**ceph-boost-1.0.1-openEuler.aarch64.rpm**
- 华为运营商网站下载
  - CentOS软件包：**ceph-boost-1.0.1-centos.aarch64.rpm**
  - openEuler软件包：**ceph-boost-1.0.1-openEuler.aarch64.rpm**

步骤2 获取软件校验工具。

- 华为企业网网站下载：**https://support.huawei.com/enterprise/zh/tool/pgp-verify-TL1000000054**
- 华为运营商网站下载：**https://support.huawei.com/carrier/digitalSignatureAction**

步骤3 使用步骤2中下载的《 OpenPGP签名验证指南 》进行软件包完整性检查。

步骤4 上传RPM包至每台服务器执行安装。

```
rpm -ivh ceph-boost-1.0.0-centos.aarch64.rpm
```

步骤5 查看服务状态。

```
systemctl status ceph-boost.service
```

```
[root@ceph1 ~]# systemctl status ceph-boost.service
● ceph-boost.service - ceph boost daemon
   Loaded: loaded (/usr/lib/systemd/system/ceph-boost.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2020-08-31 02:46:57 CST; 1 day 6h ago
 Main PID: 3968 (ceph-boost)
    Tasks: 1
   Memory: 2.1M
   CGroup: /system.slice/ceph-boost.service
           └─3968 /usr/bin/ceph-boost

Aug 31 02:46:57 ceph1 systemd[1]: Started ceph boost daemon.
```

步骤6 日志查看。

服务运行过程中生成的日志在“/var/log/ceph-boost.log”下，可通过该日志文件确定服务运行是否正常。

----结束

4 Ceph 分布式存储应用 IO 智能预取 用户指南

4.1 介绍

IO智能预取利用小容量的高速存储介质作为缓存盘，配合高效的预取算法，针对Bcache场景下，提升顺序读性能。

Ceph是一个专注于分布式的、弹性可扩展的、高可靠的、性能优异的存储系统平台。本文主要介绍如何将智能预取工具应用到Ceph集群中。

4.2 环境要求

硬件兼容列表

兼容项目	兼容性规格描述
CPU型号	华为鲲鹏920处理器
服务器型号	TaiShan 200服务器（型号2280）
	TaiShan 200服务器（型号5280）

软件兼容列表

兼容项目	兼容性规格描述
OS	CentOS Linux release 7.6.1810

兼容项目	兼容性规格描述
Ceph	Ceph 14.2.x

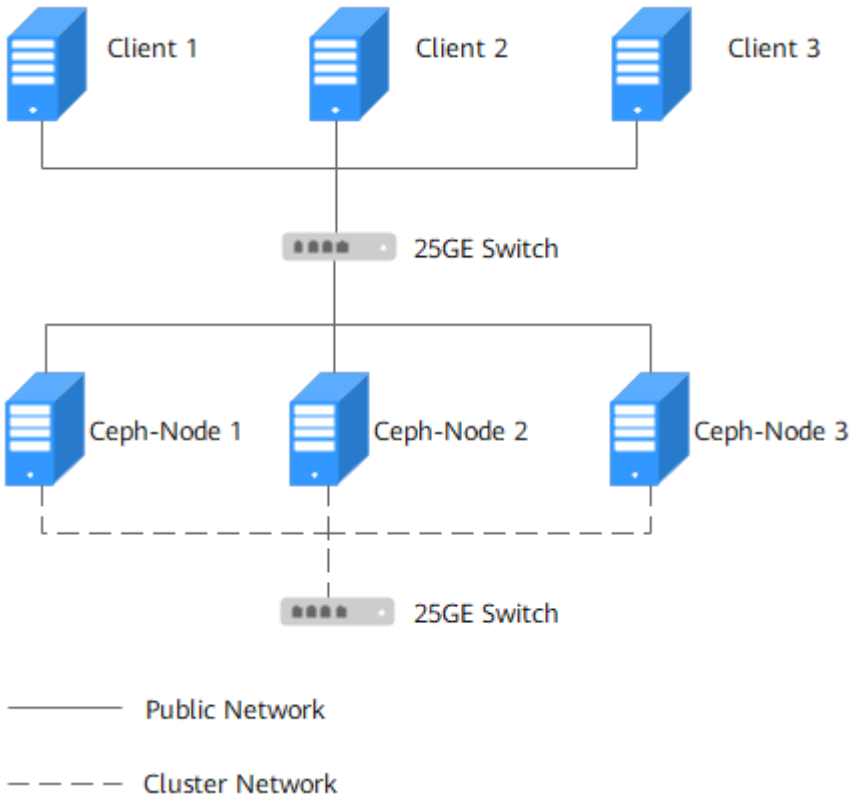
物理组网

说明

下述的环境要求，用于安装Ceph集群。

物理组网方式如图4-1所示。

图 4-1 物理组网图



集群部署过程中的IP地址规划如表4-1所示。

表 4-1 集群部署

集群节点	管理IP	Public Network	Cluster Network
Ceph-Node 1	192.168.2.166	192.168.3.166	192.168.4.166
Ceph-Node 2	192.168.2.167	192.168.3.167	192.168.4.167
Ceph-Node 3	192.168.2.168	192.168.3.168	192.168.4.168

集群部署过程中的IP地址客户端规划如表4-2所示。

表 4-2 客户端部署

客户端节点	BMC IP	Public Network
client1	192.168.2.160	192.168.3.160
client2	192.168.2.161	192.168.3.161
client3	192.168.2.162	192.168.3.162

说明

- BMC IP：用于BMC管理配置使用的IP，即带外管理IP地址。
- 内部集群IP（cluster network）：用于集群之间同步数据的IP，选取任意一个25GE网口配置即可。
- 外部访问IP（public network）：存储节点供计算节点访问的IP，用于计算节点访问块设备时连接的IP，选取任意一个25GE网口配置即可。
- 客户端当做压力机，需保证客户端与集群的前端网络在同一个网段，建议选用25GE网口进行配置。

4.3 安装 IO 智能预取工具

步骤1 确认环境中是否存在原生的Bcache模块，若存在需先卸载。

查看是否存在Bcache模块的引用计数。

```
lsmod | grep bcache
```

- 若回显如下图所示，则存在原生Bcache模块，进入步骤2。

```
[root@ceph2 sh]# lsmod | grep bcache
bcache                209305  48
You have new mail in /var/spool/mail/root
```

- 若回显如下图所示，则不存在原生Bcache模块，进入步骤3。

```
[root@node2 ceph]# lsmod | grep bcache
You have mail in /var/spool/mail/root
[root@node2 ceph]#
```

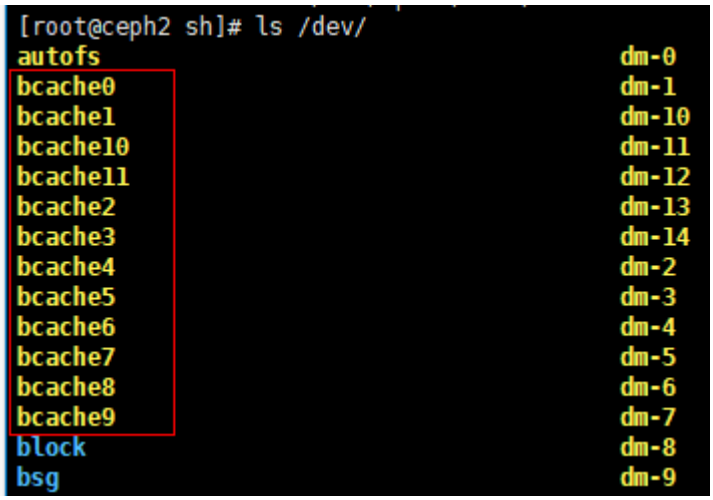
步骤2 卸载Bcache模块。

1. 卸载缓存盘。

```
echo 1 > /sys/block/[bcache]/bcache/detach
```

说明

命令中[bcache]，为“/dev”下的Bcache设备。



2. 注销缓存盘。
```
echo 1 > /sys/fs/bcache/[uuid]/unregister
```

说明

命令中具体[uuid]值需根据“/sys/fs/bcache”目录下内容进行设置，该目录下所有[uuid]设备都需要按此处理。

3. 停用后端设备。
```
echo 1 > /sys/block/[bcache]/bcache/stop
```
4. 卸载Bcache模块。
```
rmmod bcache
```

步骤3 安装Bcache模块与预取工具。

具体安装步骤请参考《**智能预取存储库 用户指南**》中的**软件编译和安装**及**配置开机自启动智能预取功能**，分别在3个Ceph集群节点安装bcache和acache_client。

----结束

4.4 使用指导

4.4.1 开启预取

步骤1 启动预取（acache_client）服务。
```
systemctl start acache_client
```

步骤2 查看预取是否开启。

1. 确认预取服务（acache_client）已经启动成功。
```
systemctl status acache_client
```
启动成功的回显如下图所示：

```
[root@cephc1 sh]# systemctl status acache_client
● acache_client.service - acache client
   Loaded: loaded (/usr/lib/systemd/system/acache_client.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2020-12-22 10:34:21 CST; 5s ago
 Main PID: 1348171 (acache_client)
   CGroup: /system.slice/acache_client.service
           └─1348171 /usr/local/bin/acache_client

Dec 22 10:34:21 cephc1 systemd[1]: Started acache client.
```

2. 查看“/dev”目录下是否存在字符设备。

   ls /dev/ | grep acache

   存在字符设备的回显如下图所示：

```
[root@cephc1 ~]# ls /dev/ | grep acache
acache
You have new mail in /var/spool/mail/root
[root@cephc1 ~]#
```

----结束

4.4.2 关闭预取

步骤1 停止预取（acache_client）服务。

systemctl stop acache_client

步骤2 确认预取服务（acache_client）已经关闭成功。

systemctl status acache_client

关闭成功的回显如下图所示：

```
[root@cephc1 ~]# systemctl status acache_client.service
● acache_client.service - acache client
   Loaded: loaded (/usr/lib/systemd/system/acache_client.service; enabled; vendor preset:
   Active: inactive (dead) since Wed 2020-12-30 14:51:39 CST; 1s ago
  Process: 9531 ExecStart=/usr/local/bin/acache_client (code=exited, status=0/SUCCESS)
 Main PID: 9531 (code=exited, status=0/SUCCESS)

Dec 30 14:51:39 cephc1 acache_client[9531]: try to read the config file : ./hcache_config
```

----结束

4.5 Ceph 使能 IO 智能预取

4.5.1 磁盘分区

Ceph14.2.10采用了BlueStore作为后端存储引擎，没有了Jewel版本的Journal盘分区的划分，而是变成DB分区（元数据分区）和WAL分区。这两个分区分别存储BlueStore后端产生的元数据和日志文件，这样整个存储系统通过元数据对数据的操作效率极高，同时通过日志事务来维持系统的稳定性。

在集群部署时，每个Ceph节点配置12块4TB数据盘和2块3.2TB的NVMe盘。每个4TB数据盘作为bcache设备的数据盘，每块NVMe盘作为6个OSD的DB、WAL分区和Bcache的cache磁盘。一般WAL分区大于10GB就足够使用，Ceph官方文档建议每个DB分区不小于每个数据盘容量的4%，其cache盘容量推荐占数据盘容量的5%-10%，具体可根据NVMe盘容量灵活设置。

在本方案中，以WAL分区设置为15GB、DB分区设置为30GB、cache盘大小为400GB（占数据盘容量10%）为例进行说明。

说明

以下操作在3个Ceph节点均执行一遍，此处以有两块NVMe（分别为/dev/nvme0n1、/dev/nvme1n1）为例进行分区，若有多块NVMe SSD，只需要在参数j中加入对应的盘符即可。若所需大小变化，更改命令end=`expr $start + 30`的数字大小为自己所需大小即可。

步骤1 创建一个partition.sh脚本。

```
vi partition.sh
```

添加如下内容：

```
#/bin/bash
for j in {0..1}
do
   parted -s /dev/nvme${j}n1  mklabel gpt
   start=0
# 划分为6个30GB分区
   end=`expr $start + 30`
   parted /dev/nvme${j}n1 mkpart primary 2048s ${end}GiB
   start=$end
   for i in {1..5}
   do
      end=`expr $start + 30`
      parted /dev/nvme${j}n1 mkpart primary ${start}GiB ${end}GiB
      start=$end
   done
# 划分为6个15GB分区
   for i in {1..6}
   do
      end=`expr $start + 15`
      parted /dev/nvme${j}n1 mkpart primary ${start}GiB ${end}GiB
      start=$end
   done
# 划分为6个400GB分区
   for i in {1..6}
   do
      end=`expr $start + 400`
      parted /dev/nvme${j}n1 mkpart primary ${start}GiB ${end}GiB
      start=$end
   done
done
```

说明

此脚本内容只适用于当前硬件配置，其他硬件配置可参考此脚本。

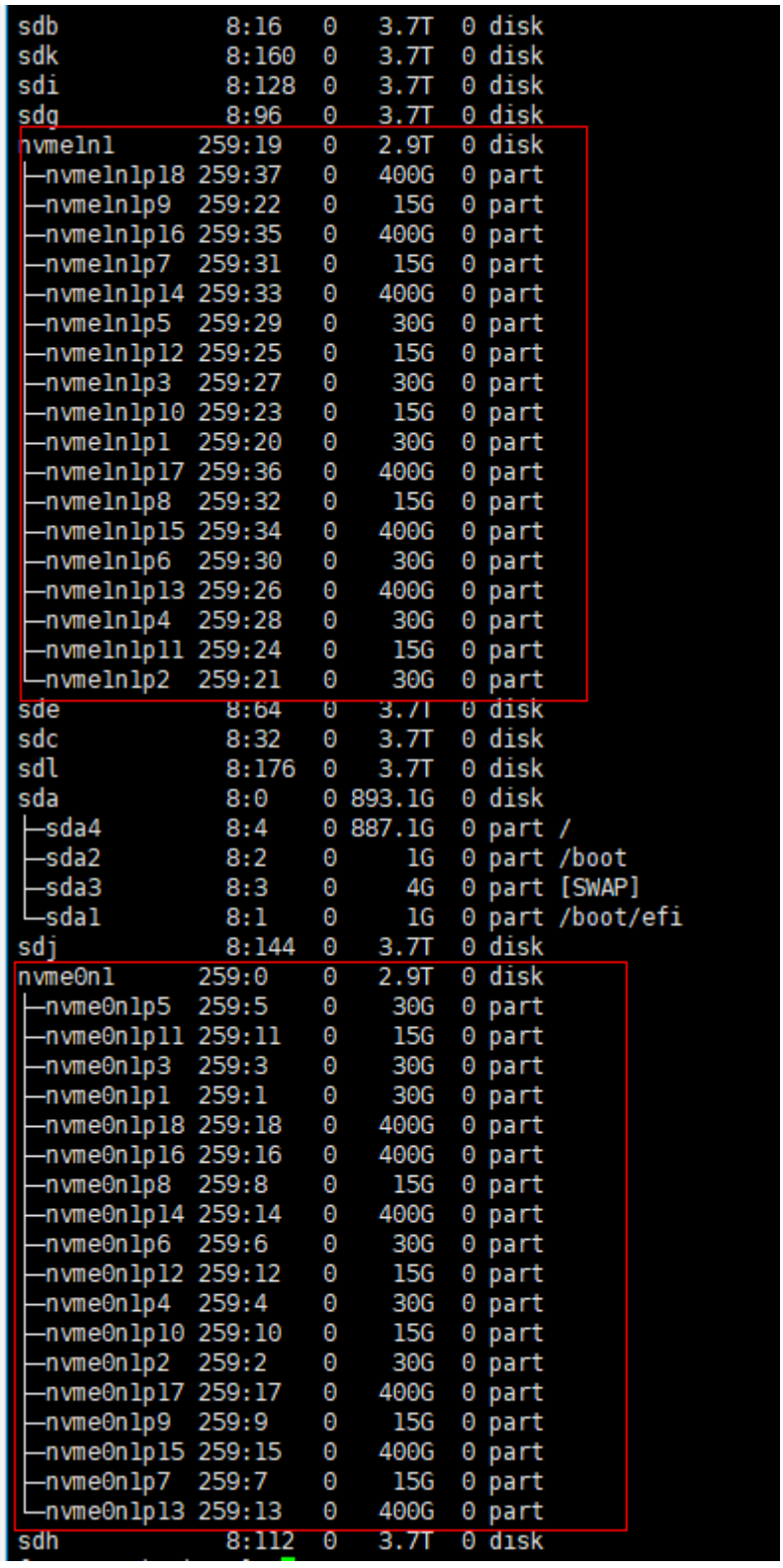
步骤2 执行脚本。

```
bash partition.sh
```

步骤3 查看分区是否创建成功。

```
lsblk
```

创建成功的回显如下图所示：



----结束

4.5.2 创建 Bcache 设备

说明

Bcache盘分为数据盘和cache盘，一般采用HDD作为数据盘，SDD作为cache盘。以下操作在3个ceph节点均执行一遍，脚本中的/dev/sda-/dev/sdl 12块硬盘均为Bcache设备的数据盘，将NVMe中分区大小为400G的作为Cache盘。本例中以/dev/nvme0n1p$n为例，n的取值为{13..18}，这里的数值与上述分区之后的数值对应。

注意

实际情况中可能会遇到OS硬盘位于HDD盘中的情况，例如系统盘安装到了/dev/sda，则不能直接使用以下脚本直接运行，否则部署到make-bcache --wipe-bcache -B /dev/sda时会报错。此时需要重新调整脚本，避免脚本中包含数据盘以外的如OS盘、做DB/WAL分区的SSD盘等。

因此在操作前先查看磁盘分区情况。

```
lsblk
```

如图所示，sda盘为系统盘。

```
sda             8:0    0 893.1G  0 disk
├─sda4          8:4    0 887.1G  0 part /
├─sda2          8:2    0     1G  0 part /boot
├─sda3          8:3    0     4G  0 part [SWAP]
└─sda1          8:1    0     1G  0 part /boot/efi
```

步骤1 创建一个create_bcache.sh脚本。

```
vi create_bcache.sh
```

添加如下内容：

```
#!/bin/bash
n=13
for disk in {a..f}
do
      make-bcache -B /dev/sd${disk} -C /dev/nvme0n1p${n}
      ((n = $(( $n + 1 ))))
done
n=13
for disk in {g..l}
do
      make-bcache -B /dev/sd${disk} -C /dev/nvme1n1p$n
      ((n = $(($n + 1))))
done
```

说明

make-bcache -B /dev/sd${disk} -C /dev/nvme0n1p${n}中的参数含义为：

- -B：指定后端磁盘设备，即数据盘。
- -C：指定缓存设备，用于加速数据盘。

举例：后端磁盘为“sdb”，缓存设备为“nvme0n1p13”

make-bcache -B /dev/sdb -C /dev/nvme0n1p13

步骤2 执行脚本。

```
bash create_bcache.sh
```

步骤3 查看Bcache设备是否创建成功。

```
lsblk
```

```
[root@cephc1 zl]# lsblk
NAME            MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sdf               8:80   0   3.7T  0 disk
└─bcache80      253:80   0   3.7T  0 disk
sdd               8:48   0   3.7T  0 disk
└─bcache48      253:48   0   3.7T  0 disk
sdm               8:192  0   1.1T  0 disk
├─sdm3            8:195  0   1.1T  0 part
  ├─centos-swap 252:1    0     4G  0 lvm
  ├─centos-home 252:2    0     1T  0 lvm  /home
  └─centos-root 252:0    0    50G  0 lvm  /
├─sdm1            8:193  0   200M  0 part /boot/efi
└─sdm2            8:194  0     1G  0 part /boot
sdb               8:16   0   3.7T  0 disk
└─bcache16      253:16   0   3.7T  0 disk
sdk               8:160  0   3.7T  0 disk
└─bcache160     253:160  0   3.7T  0 disk
sdi               8:128  0   3.7T  0 disk
└─bcache128     253:128  0   3.7T  0 disk
sdg               8:96   0   3.7T  0 disk
└─bcache96      253:96   0   3.7T  0 disk
nvme1n1         259:0    0   2.9T  0 disk
├─nvme1n1p18    259:37   0 372.5G  0 part
  └─bcache176   253:176  0   3.7T  0 disk
├─nvme1n1p9     259:28   0    28G  0 part
├─nvme1n1p16    259:35   0 372.5G  0 part
  └─bcache144   253:144  0   3.7T  0 disk
├─nvme1n1p7     259:26   0    28G  0 part
├─nvme1n1p14    259:33   0 372.5G  0 part
  └─bcache112   253:112  0   3.7T  0 disk
├─nvme1n1p5     259:24   0    14G  0 part
├─nvme1n1p12    259:31   0    28G  0 part
├─nvme1n1p3     259:22   0    14G  0 part
├─nvme1n1p10    259:29   0    28G  0 part
├─nvme1n1p1     259:20   0    14G  0 part
├─nvme1n1p17    259:36   0 372.5G  0 part
  └─bcache160   253:160  0   3.7T  0 disk
├─nvme1n1p8     259:27   0    28G  0 part
├─nvme1n1p15    259:34   0 372.5G  0 part
  └─bcache128   253:128  0   3.7T  0 disk
├─nvme1n1p6     259:25   0    14G  0 part
├─nvme1n1p13    259:32   0 372.5G  0 part
  └─bcache96    253:96   0   3.7T  0 disk
├─nvme1n1p4     259:23   0    14G  0 part
├─nvme1n1p11    259:30   0    28G  0 part
└─nvme1n1p2     259:21   0    14G  0 part
sde               8:64   0   3.7T  0 disk
└─bcache64      253:64   0   3.7T  0 disk
sdc               8:32   0   3.7T  0 disk
└─bcache32      253:32   0   3.7T  0 disk
sdl               8:176  0   3.7T  0 disk
└─bcache176     253:176  0   3.7T  0 disk
sda               8:0    0   3.7T  0 disk
└─bcache0       253:0    0   3.7T  0 disk
```

Bcache设备可找到对应的数据盘和相对应的cache盘，则创建成功。

----结束

4.5.3 Ceph 部署

步骤1 安装Ceph软件并部署MON、MGR节点。

详细操作请参考《**Ceph块存储 部署指南（CentOS 7.6）**》中的**安装Ceph软件**、**部署MON节点**和**部署MGR节点**相关内容。

步骤2 部署OSD节点。

说明

操作前请确认哪些硬盘作为数据盘使用，并确保数据盘中没有未清理的分区。若存在未清理分区，需先进行清除。

1. 查看各硬盘下是否有分区。
   ```
   lsblk
   ```
2. 若存在分区信息，则清除分区信息（以盘符/dev/sdb为例）。
   ```
   ceph-volume lvm zap /dev/sdb --destroy
   ```

1. 在Ceph-Node 1上创建脚本create_osd.sh，将每台服务器上的12块Bcache盘作为OSD的数据盘。
   ```
   cd /etc/ceph
   vi /etc/ceph/create_osd.sh
   ```

添加以下内容：

```
#!/bin/bash
for node in ceph1 ceph2 ceph3
do
        j=7
        k=1
        for i in `ssh ${node} "ls /sys/block | grep bcache | head -n 6"`
        do
                ceph-deploy osd create ${node} --data /dev/${i} --block-wal /dev/nvme0n1p${j} --block-db /dev/nvme0n1p${k}
                ((j=${j}+1))
                ((k=${k}+1))
                sleep 3
        done
        j=7
        k=1
        for i in `ssh ${node} "ls /sys/block | grep bcache | tail -n 6"`
        do
                ceph-deploy osd create ${node} --data /dev/${i} --block-wal /dev/nvme1n1p${j} --block-db /dev/nvme1n1p${k}
                ((j=${j}+1))
                ((k=${k}+1))
                sleep 3
        done
done
```

说明

- 此脚本内容只适用于当前硬件配置，其他硬件配置可参考此脚本。
- **ceph-deploy osd create**命令中:
  - ${node}是节点的hostname。
  - --data选项后面是作为数据盘的设备，以Bcache的后端盘作为数据盘。
  - --block-db选项后面是DB分区。
  - --block-wal选项后面是WAL分区。
- DB和WAL通常部署在NVMe SSD上以提高写入性能，如果没有配置NVMe SSD或者直接使用NVMe SSD作为数据盘，则不需要和--block-wal，只需要加--data指定数据盘即可。

2. 在ceph1上运行脚本。

```
bash create_osd.sh
```

3. 创建成功后，查看OSD是否创建成功。

```
ceph -s
```

```
[root@ceph2 ~]# ceph -s
  cluster:
    id:     7f113d15-c4cf-4ee0-a72b-f3170808d2ea
    health: HEALTH_OK

  services:
    mon: 3 daemons, quorum ceph1,ceph2,ceph3 (age 103m)
    mgr: ceph1(active, since 2h), standbys: ceph3, ceph2
    osd: 36 osds: 36 up (since 89m), 36 in (since 90m)

  data:
```

36个OSD都为up即为创建成功。

----结束

4.5.4 验证 Ceph

开启IO智能预取之后，正常使用Ceph即可在Ceph中运行IO智能预取的功能。

Ceph的使用和验证指导请参考《**Ceph块存储 部署指南（CentOS 7.6）**》中**验证Ceph**的相关内容。

A 修订记录

发布日期	修订记录
2021-06-24	第七次正式发布。 《1 Bcache 用户指南（CentOS 7.6）》和《2 Bcache 用户指南（openEuler 20.03）》中删除构建bcache-tools RPM包相关内容。
2021-05-26	第六次正式发布。 特性智能预取更名为IO智能预取。
2021-03-23	第五次正式发布。 鲲鹏分布式存储解决方案更名为鲲鹏BoostKit分布式存储使能套件。
2021-01-27	第四次正式发布。 新增《4 Ceph分布式存储应用IO智能预取 用户指南》。
2020-11-30	第三次正式发布。 新增《3 IO直通工具 用户指南》。
2020-09-27	第二次正式发布。 原《Bcache 用户指南》基于不同的操作系统拆分成两本：《1 Bcache 用户指南（CentOS 7.6）》和《2 Bcache 用户指南（openEuler 20.03）》。
2020-6-24	第一次正式发布。

