Kunpeng BoostKit
23.0.0
分布式存储加速算法库

文档版本 01
发布日期 2024-01-11

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暗示的声明或保证。

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。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： https://www.huawei.com

客户服务邮箱： support@huawei.com

客户服务电话： 4008302118

安全声明

漏洞处理流程

华为公司对产品漏洞管理的规定以“漏洞处理流程”为准，该流程的详细内容请参见如下网址：
https://www.huawei.com/cn/psirt/vul-response-process
如企业客户须获取漏洞信息，请参见如下网址：
https://securitybulletin.huawei.com/enterprise/cn/security-advisory

目 录

1 介绍

存储加速算法库（简称KSAL）是华为自研的存储算法加速库，当前包括EC算法、CRC16 T10DIF算法和CRC32C算法。本文用于指导用户安装并使能KSAL算法加速库。

简要介绍

5G与AI等新兴技术的发展，加速了数据的产生和流动。根据华为全球产业展望GIV（Global Industry Vision）报告的数据，到2025年全球数据量可达到180ZB。多样性业务正带来数据前所未有的增长，数据也变得越来越重要。随着数据量增多，应用对于存储系统的性能要求不断提高。面对新兴应用越来越高的性能诉求，如何不断提高存储系统性能以满足业务需求，成为一大挑战。

KSAL（Kunpeng Storage Acceleration Library）是华为自研的存储算法加速库，当前包括EC（Erasure Code）算法、CRC16 T10DIF（Cyclic Redundancy Check 16 T10 Data Integrity Field）算法和CRC32C（Cyclic Redundancy Check 32 Castagnoli）算法。

EC算法基于华为自研向量化EC编解码方案，通过同构映射将EC编码过程中所需的高阶有限域GF(2^w)乘法操作替换为二元矩阵乘法，进而将查表实现的复杂有限域乘法操作替代为XOR（Exclusive OR）操作，同时采用编码编排算法在校验块计算过程中对中间结果进行复用，减少XOR操作数，配合鲲鹏向量化指令实现编码加速。相比开源EC算法，KSAL EC算法性能更好。与主流开源EC算法相比，编码性能提升1倍以上。

CRC16 T10DIF算法和CRC32C算法，通过大数求余算法和配合鲲鹏向量化指令实现编码加速，相比开源算法，CRC16 T10DIF算法4K性能提升30%，CRC32C算法4K性能提升130%。

本文主要介绍了KSAL软件的获取、安装、部署、验证、使用以及在Ceph上的使能KSAL EC算法的方法。

2 约束与限制

KSAL目前针对处理器型号和EC算法配置存在一定约束。

KSAL存在以下约束：

- KSAL算法加速库仅支持华为鲲鹏920处理器使用。
- 当前EC算法仅支持2+2、4+2、6+2、12+3和20+3配比，其他配比暂不支持。

3 环境要求

本文基于TaiShan服务器和openEuler操作系统提供指导，在正式操作前请确保软硬件均满足要求。

硬件要求

硬件规格表3-1所示。

表 3-1 硬件要求

项目	规格
CPU型号	华为鲲鹏920处理器
服务器	TaiShan 200服务器（型号2280）
	TaiShan 200服务器（型号5280）

操作系统和软件要求

操作系统和软件要求如表3-2所示。

表 3-2 操作系统和软件要求

项目	版本
操作系统	openEuler 20.03 LTS SP1
	openEuler 22.03 LTS SP1
Ceph	14.2.8

说明

由于KSAL是华为自研闭源算法库，KSAL仅支持华为鲲鹏处理器使用。

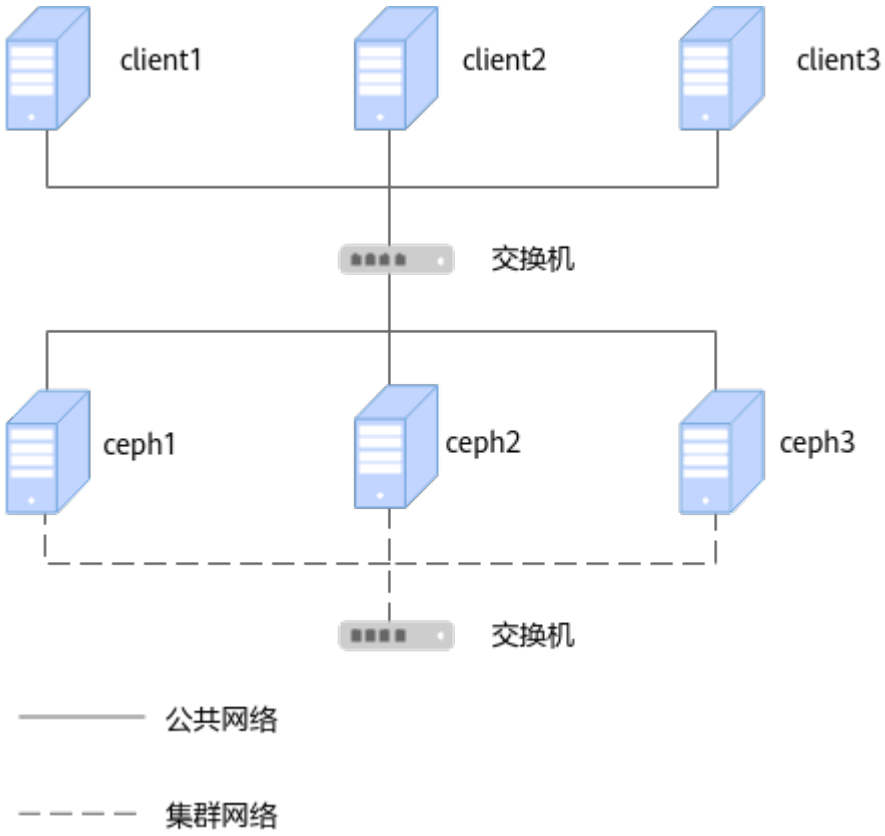
集群规划

本文以在Ceph中使能KSAL EC算法为例，集群规划为三节点集群，包含三台Ceph服务端服务器和三台Ceph客户端服务器。若用户需要直接调用KSAL算法加速库，进行系统性能优化，可根据实际情况进行集群规划。

在Ceph软件编译过程中，需要使用到一台服务器作为编译机，可根据实际情况，在集群之外单独使用一台服务器作为编译机，或使用集群中的一台客户端服务器作为编译机。

集群组网图如图3-1所示。

图 3-1 集群规划



集群规划各节点详情如表3-3所示。

表 3-3 集群节点

节点名称	节点角色	公共网络IP地址示例	集群网络IP地址示例
ceph1	服务端	192.168.3.166	192.168.4.166
ceph2	服务端	192.168.3.167	192.168.4.167
ceph3	服务端	192.168.3.168	192.168.4.168
client1	客户端	192.168.3.160	NA

节点名称	节点角色	公共网络IP地址示例	集群网络IP地址示例
client2	客户端	192.168.3.161	NA
client3	客户端	192.168.3.162	NA

4 获取软件

4.1 获取软件包

在执行编译、安装、部署之前，需要分别从开源网站和鲲鹏社区获取相关软件包。

从开源网站获取

安装包名称	说明	获取路径
ceph-14.2.8.tar.gz	Ceph源码包，用于编译机编译出包，后续用于服务端和客户端安装。	**下载地址**
ceph-ksal-ec-plugin.patch	KSAL EC算法插件，用于编译机编译出包，后续用于服务端和客户端安装。	**下载地址**

从鲲鹏社区获取

须知

使用软件包前请先阅读**《鲲鹏应用使能套件BoostKit用户许可协议 2.0》**，如确认继续使用，则默认同意协议的条款和条件。

软件包名称	说明	获取途径
BoostKit-KSAL_1.1.0.zip	KSAL闭源算法包，用于服务端和客户端安装。	下载地址

4.2 完整性校验

获取软件包后，需要校验软件包，确保与网站上的原始软件包一致。

操作步骤

步骤1 从上面步骤官方的企业网或运营商网站获取对应的软件数字证书和软件安装包。

步骤2 获取校验工具和校验方法。

https://support.huawei.com/enterprise/zh/tool/pgp-verify-TL1000000054

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

----结束

5 安装 KSAL 闭源算法包

请在集群所有节点安装KSAL。KSAL通过RPM包形式安装，解压zip包获取RPM包直接安装即可。

操作步骤

步骤1 获取BoostKit-KSAL_1.1.0.zip，放置于“/home”目录下。

步骤2 在“/home”目录下面解压BoostKit-KSAL_1.1.0.zip。

```
unzip BoostKit-KSAL_1.1.0 .zip
```

步骤3 安装解压的RPM包。

```
rpm -ivh /home/libksal-release-1.1.0.oe1.aarch64.rpm
```

步骤4 确认RPM安装情况。

```
rpm -qi libksal
```

显示如下：

```
Name        : libksal
Version     : 1.1.0
Release     : 1.el7
Architecture: aarch64
Install Date: Wed 03 Jan 2024 07:44:14 PM CST
Group       : Unspecified
Size        : 134366
License     : GPL
Signature   : (none)
Source RPM  : libksal-1.1.0-1.el7.src.rpm
Build Date  : Thu 16 Nov 2023 09:45:14 AM CST
Build Host  : ceph1
Relocations : (not relocatable)
Summary     : Kunpeng Storage Acceleration Library
Description :
Kunpeng Storage Acceleration Library
```

----结束

后续操作

KSAL闭源算法包安装完成后，若需要在Ceph中使能KSAL EC算法，请参见**6 编译部署Ceph**完成Ceph集群的部署和使能KSAL EC算法的操作。若需要直接调用KSAL算法加速库，进行系统性能优化，则可跳过**6 编译部署Ceph**，参见**7 KSAL开发参考**进行算法调用。

6 编译部署 Ceph

6.1 Ceph 源码合入 KSAL EC 算法插件

执行编译之前，需要在编译机上将KSAL EC算法插件合入Ceph源码，使Ceph支持KSAL EC算法。请任意选择一台客户端节点作为编译机。

步骤1 获取ceph-14.2.8.tar.gz和ceph-ksal-ec-plugin.patch，放置于“/home”目录下。

步骤2 对源码进行解压。

```
tar zxvf ceph-14.2.8.tar.gz
```

步骤3 把ceph-ksal-ec-plugin.patch放入“/home/ceph-14.2.8”目录下。

```
cp /home/ceph-ksal-ec-plugin.patch /home/ceph-14.2.8
```

步骤4 合入插件。

```
cd /home/ceph-14.2.8
patch -p1 < ceph-ksal-ec-plugin.patch
```

----结束

6.2 编译 Ceph 安装包

源码合入插件后需编译生成新的Ceph安装包，主要包括配置编译环境、修改相关文件和执行编译等操作。请在编译机上完成编译出包的操作。

前提条件

编译时“/home”目录下需要存放编译的中间结果和目标文件，建议“/home”目录下的容量不小于100GB。

配置本地源

步骤1 下载liboath源码及补丁。

```
yum install git -y
git config --global http.sslVerify false
git clone https://gitee.com/src-openeuler/oath-toolkit.git
```

步骤2 通过Yum方式安装RPM打包所需的依赖。

```
yum install wget rpmdevtools gtk-doc pam-devel xmlsec1-devel libtool libtool-ltdl-devel createrepo cmake -y
```

步骤3 创建“rpmbuild”目录，并将patch文件和源码包移动到“/root/rpmbuild/SOURCES”目录下。

```
rpmdev-setuptree
cd oath-toolkit
mv 0001-oath-toolkit-2.6.5-lockfile.patch /root/rpmbuild/SOURCES
mv oath-toolkit-2.6.5.tar.gz /root/rpmbuild/SOURCES
cp oath-toolkit.spec /root/rpmbuild/SPECS/
```

步骤4 编译RPM包。

```
rpmbuild -bb /root/rpmbuild/SPECS/oath-toolkit.spec
```

步骤5 将编译好的RPM包作为本地Yum源。

```
mkdir -p /home/oath
cp -r /root/rpmbuild/RPMS/*  /home/oath/
cd  /home/oath && createrepo .
```

步骤6 配置repo文件。

1. 新建“local.repo”文件。
   ```
   vi /etc/yum.repos.d/local.repo
   ```
2. 按“i”键进入编辑模式，在文件中加入以下内容。
   ```
   [local-oath]
   name=local-oath
   baseurl=file:///home/oath
   enabled=1
   gpgcheck=0
   priority=1
   ```
3. 按“Esc”键退出编辑模式，输入**:wq!**并按“Enter”键保存退出文件。

步骤7 设置Yum证书验证状态为不验证。

1. 打开“yum.conf”文件。
   ```
   vim /etc/yum.conf
   ```
2. 按“i”键进入编辑模式，添加如下内容至末尾。
   ```
   sslverify=false
   deltarpm=0
   ```
3. 按“Esc”键退出编辑模式，输入**:wq!**并按“Enter”键保存退出文件。

步骤8 将pip源配置为华为镜像源，以提高下载速度。

1. 新建“.pip”目录并在该目录下新建“pip.conf”文件。
   ```
   mkdir -p ~/.pip
   vim ~/.pip/pip.conf
   ```
2. 按“i”键进入编辑模式，添加如下内容。
   ```
   [global]
   timeout = 120
   ```



```
index-url = https://repo.huaweicloud.com/repository/pypi/simple
trusted-host = repo.huaweicloud.com
```

3. 按“Esc”键退出编辑模式，输入**:wq!**并按“Enter”键保存退出文件。

----结束

安装依赖包

步骤1 配置epel源。

1. 新建“epel.repo”文件。
   ```
   vim /etc/yum.repos.d/epel.repo
   ```
2. 按“i”键进入编辑模式，添加如下内容。
   ```
   [epel]
   name=epel
   baseurl=https://repo.huaweicloud.com/epel/7/aarch64/
   enabled=1
   gpgcheck=0
   priority=1
   ```
3. 按“Esc”键退出编辑模式，输入**:wq!**并按“Enter”键保存退出文件。

步骤2 使用epel源安装openEuler中缺少的依赖。

```
yum install python-routes python-tox -y
```

步骤3 删除epel源。

```
rm -rf /etc/yum.repos.d/epel.repo
```

须知

epel源仅用于**步骤2**中的两个依赖的安装，使用后必须删除，否则后续步骤将会从epel源下载与openEuler冲突的RPM包。

步骤4 下载华为镜像源repo。

```
wget -O /etc/yum.repos.d/openEulerOS.repo https://repo.huaweicloud.com/repository/conf/openeuler_aarch64.repo
```

步骤5 安装服务端Ceph源码编译需要的依赖。

```
yum install java-devel sharutils checkpolicy selinux-policy-devel gperf cryptsetup fuse-devel gperftools-devel libaio-devel libblkid-devel libcurl-devel libudev-devel libxml2-devel libuuid-devel ncurses-devel python-devel valgrind-devel xfsprogs-devel xmlstarlet yasm nss-devel libibverbs-devel openldap-devel CUnit-devel python2-Cython python3-setuptools python-prettytable lttng-ust-devel expat-devel junit boost-random keyutils-libs-devel openssl-devel libcap-ng-devel python-sphinx python2-sphinx python3-sphinx leveldb leveldb-devel snappy snappy-devel lz4 lz4-devel liboath liboath-devel libbabeltrace-devel librabbitmq librabbitmq-devel librdkafka librdkafka-devel libnl3 libnl3-devel rdma-core-devel numactl numactl-devel numactl-libs createrepo openldap-devel rdma-core-devel lz4-devel expat-devel lttng-ust-devel libbabeltrace-devel python3-Cython python2-Cython gperftools-devel bc dnf-plugins-core librabbitmq-devel rpm-build java-1.8.0-openjdk-devel -y
```

----结束

编译出包

步骤1 修改“install-deps.sh”脚本。

1. 打开脚本文件。
   ```
   cd /home/ceph-14.2.8/
   vi install-deps.sh
   ```


2. 按“i”键进入编辑模式，如下所示在文件321行处新增“openEuler”。
   ```
   centos|fedora|rhel|ol|virtuozzo|openEuler)
   ```
3. 按“Esc”键退出编辑模式，输入**:wq!**并按“Enter”键保存退出文件。

步骤2 运行“install-deps.sh”，安装依赖包。

```
sh install-deps.sh
```

说明

install-deps.sh脚本从Ceph源码包中获取。如果执行失败，可以尝试切换代理后重新运行。openEuler 20.03 LTS SP3版本操作系统存在两个版本的PostgreSQL，安装时会冲突，为避免冲突时脚本退出，需要删除“install-deps.sh”中的**set -e**后执行脚本。

步骤3 修改ceph.spec.in文件。

```
sed -i 's#%if 0%{?fedora} || 0%{?rhel}#%if 0%{?fedora} || 0%{?rhel} || 0%{?openEuler}#' ceph.spec.in
```

步骤4 回到上级目录并将ceph-14.2.8目录打包为tar.bz2格式的压缩包。

```
cd /home
tar -cjvf ceph-14.2.8.tar.bz2 ceph-14.2.8
```

步骤5 修改rpmmacros文件。

1. 打开文件。
   ```
   vi /root/.rpmmacros
   ```
2. 按“i”进入编辑模式，修改“%_topdir”的路径为“/home/rpmbuild”，并注释掉如下四行代码。
   ```
   %_topdir /home/rpmbuild

   #%__arch_install_post \
   #    [ "%{buildarch}" = "noarch" ] || QA_CHECK_RPATHS=1 ; \
   #    case "${QA_CHECK_RPATHS:-}" in [1yY]*) /usr/lib/rpm/check-rpaths ;; esac \
   #    /usr/lib/rpm/check-buildroot
   ```
3. 按“Esc”键退出编辑模式，输入**:wq!**并按“Enter”键保存退出文件。
4. 再次执行rpmbuild安装命令。
   ```
   rpmdev-setuptree
   ```

步骤6 将ceph.spec文件拷贝到SPECS目录下。

```
cp ceph-14.2.8/ceph.spec /home/rpmbuild/SPECS/
```

步骤7 将**步骤4**打包好的文件放到SOURCES目录下。

```
cp ceph-14.2.8.tar.bz2 /home/rpmbuild/SOURCES/
```

步骤8 删除“/etc/profile.d/performance.sh”以提高编译速度。

```
rm -rf  /etc/profile.d/performance.sh
```

步骤9 重新开启一个新的终端让**步骤8**生效，使用rpmbuild开始编译。

```
unset GOMP_CPU_AFFINITY
rpmbuild -bb /home/rpmbuild/SPECS/ceph.spec
```

编译过程需要花费半小时以上，编译完成后会在“/home/rpmbuild/RPMS/”目录下生成两个目录“aarch64”和“noarch”，其中包含有**部署Ceph集群**相关的RPM包。

```
[root@localhost RPMS]# ls
aarch64 noarch
[root@localhost RPMS]# ls *
aarch64:
ceph-14.2.8-0.aarch64.rpm              Librbd-devel-14.2.8-0.aarch64.rpm
ceph-base-14.2.8-0.aarch64.rpm         librgw2-14.2.8.0.aarch64.rpm
ceph-common-14.2.8-0.aarch64.rpm       librgw-devel-14.2.8-0.aarch64.rpm
ceph-debuginfo-14.2.8-0.aarch64.rpm    python3-ceph-argparse-14.2.8-0.aarch64.rpm
ceph-debugsource-14.2.8-0.aarch64.rpm  python3-cephfs-14.2.8-0.aarch64.rpm
ceph-fuse-14.2.8-0.aarch64.rpm         python3-rados-14.2.8-0.aarch64.rpm
```



```
ceph-nds-14.2.8-0.aarch64.rpm            python3-rbd-14.2.8-0.aarch64.rpm
ceph-mgr-14.2.8-0.aarch64.rpm            python3-rgw-14.2.8-0.aarch64.rpm
ceph-mon-14.2.8-0.aarch64.rpm            python-ceph-argparse-14.2.8-0.aarch64.rpm
ceph-osd-14.2.8-0.aarch64.rpm            python-ceph-compat-14.2.8-0.aarch64.rpm
ceph-radosgw-14.2.8-0.aarch64.rpm        python-cephfs-14.2.8-0.aarch64.rpm
ceph-test-14.2.8-0.aarch64.rpm           python-rados-14.2.8-0.aarch64.rpm
Libcephfs2-14.2.8-0.aarch64.rpm          python-rbd-14.2.8-0.aarch64.rpm
Libcephfs-devel-14.2.8-0.aarch64.rpm     python-rgw-14.2.8-0.aarch64.rpm
Librados2-14.2.8-0.aarch64.rpm           rados-objclass-devel-14.2.8-0.aarch64.rpm
librados-devel-14.2.8-0.aarch64.rpm      rbd-fuse-14.2.8-0.aarch64.rpm
libradospp-devel-14.2.8-0.aarch64.rpm    rbd-mirror-14.2.8-0.aarch64.rpm
librbd1-14.2.8-0.aarch64.rpm             rbd-nbd-14.2.8-0.aarch64.rpm
noarch:
ceph-grafana-dashboards-14.2.8-0.noarch.rpm         ceph-mgr-k8sevents-14.2.8-0.noarch.rpm
ceph-mgr-dashboard-14.2.8-0.noarch.rpm              ceph-mgr-rook-14.2.8-0.noarch.rpm
ceph-mgr-diskprediction-cloud-14.2.8-0.noarch.rpm   ceph-mgr-ssh-14.2.8-0.noarch.rpm
ceph-mgr-diskprediction-local-14.2.8-0.noarch.rpm
```

步骤10 打包编译出的RPM包。

```
cd /home/rpmbuild/RPMS
tar -zcvf ceph-ksal-rpm.tar.gz aarch64/ noarch/
```

----结束

6.3 部署 Ceph 集群

6.3.1 配置部署环境

安装部署前需进行环境配置，主要包括配置下载源、关闭防火墙、配置主机名、配置NTP（Network Time Protocol）、配置免密登录和关闭SELinux等操作。

配置下载源

步骤1 在所有服务端节点确认openEuler源。

1. 确认“openEuler.repo”文件和openEuler源对应。

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

   - openEuler 20.03源如下所示。

     ```
     [OS]
     name=OS
     baseurl=http://repo.openeuler.org/openEuler-20.03-LTS-SP1/OS/$basearch/
     enabled=1
     gpgcheck=0
     gpgkey=http://repo.openeuler.org/openEuler-20.03-LTS-SP1/OS/$basearch/RPM-GPG-KEY-openEuler

     [everything]
     name=everything
     baseurl=http://repo.openeuler.org/openEuler-20.03-LTS-SP1/everything/$basearch/
     enabled=1
     gpgcheck=0
     gpgkey=http://repo.openeuler.org/openEuler-20.03-LTS-SP1/everything/$basearch/RPM-GPG-KEY-openEuler

     [EPOL]
     name=EPOL
     baseurl=http://repo.openeuler.org/openEuler-20.03-LTS-SP1/EPOL/$basearch/
     enabled=1
     gpgcheck=0
     gpgkey=http://repo.openeuler.org/openEuler-20.03-LTS-SP1/OS/$basearch/RPM-GPG-KEY-openEuler

     [debuginfo]
     ```



```
name=debuginfo
baseurl=http://repo.openeuler.org/openEuler-20.03-LTS-SP1/debuginfo/$basearch/
enabled=1
gpgcheck=0
gpgkey=http://repo.openeuler.org/openEuler-20.03-LTS-SP1/debuginfo/$basearch/RPM-GPG-KEY-openEuler

[source]
name=source
baseurl=http://repo.openeuler.org/openEuler-20.03-LTS-SP1/source/
enabled=1
gpgcheck=0
gpgkey=http://repo.openeuler.org/openEuler-20.03-LTS-SP1/source/RPM-GPG-KEY-openEuler

[update]
name=update
baseurl=http://repo.openeuler.org/openEuler-20.03-LTS-SP1/update/$basearch/
enabled=1
gpgcheck=0
gpgkey=http://repo.openeuler.org/openEuler-20.03-LTS-SP1/OS/$basearch/RPM-GPG-KEY-openEuler
```

- openEuler 22.03源如下所示。

```
[OS]
name=OS
baseurl=http://repo.openeuler.org/openEuler-22.03-LTS-SP1/OS/$basearch/
enabled=1
gpgcheck=1
gpgkey=http://repo.openeuler.org/openEuler-22.03-LTS-SP1/OS/$basearch/RPM-GPG-KEY-openEuler

[everything]
name=everything
baseurl=http://repo.openeuler.org/openEuler-22.03-LTS-SP1/everything/$basearch/
enabled=1
gpgcheck=1
gpgkey=http://repo.openeuler.org/openEuler-22.03-LTS-SP1/everything/$basearch/RPM-GPG-KEY-openEuler

[EPOL]
name=EPOL
baseurl=http://repo.openeuler.org/openEuler-22.03-LTS-SP1/EPOL/main/$basearch/
enabled=1
gpgcheck=1
gpgkey=http://repo.openeuler.org/openEuler-22.03-LTS-SP1/OS/$basearch/RPM-GPG-KEY-openEuler

[debuginfo]
name=debuginfo
baseurl=http://repo.openeuler.org/openEuler-22.03-LTS-SP1/debuginfo/$basearch/
enabled=1
gpgcheck=1
gpgkey=http://repo.openeuler.org/openEuler-22.03-LTS-SP1/debuginfo/$basearch/RPM-GPG-KEY-openEuler

[source]
name=source
baseurl=http://repo.openeuler.org/openEuler-22.03-LTS-SP1/source/
enabled=1
gpgcheck=1
gpgkey=http://repo.openeuler.org/openEuler-22.03-LTS-SP1/source/RPM-GPG-KEY-openEuler

[update]
name=update
baseurl=http://repo.openeuler.org/openEuler-22.03-LTS-SP1/update/$basearch/
enabled=1
gpgcheck=1
gpgkey=http://repo.openeuler.org/openEuler-22.03-LTS-SP1/OS/$basearch/RPM-GPG-KEY-openEuler
```



```
[update-source]
name=update-source
baseurl=http://repo.openeuler.org/openEuler-22.03-LTS-SP1/update/source/
enabled=1
gpgcheck=1
gpgkey=http://repo.openeuler.org/openEuler-22.03-LTS-SP1/source/RPM-GPG-KEY-openEuler
```

2. 确认无误后输入**:q!**并按“Enter”键退出文件。

步骤2 在所有节点将**6.2 编译Ceph安装包**编译出的使能KSAL后的Ceph RPM包配置为本地源。

1. 创建并进入“ceph-ksal”路径。
   ```
   mkdir -p /home/ceph-ksal
   cd /home/ceph-ksal
   ```
2. 将**6.2 编译Ceph安装包**编译出RPM包放置在“ceph-ksal”路径下并解压。
   ```
   cp /home/rpmbuild/RPMS/ceph-ksal-rpm.tar.gz /home/ceph-ksal
   tar -zxvf ceph-ksal-rpm.tar.gz
   ```
3. 创建本地源。
   ```
   createrepo .
   ```
4. 打开“local.repo”文件。
   ```
   vi /etc/yum.repos.d/local.repo
   ```
5. 按“i”进入编辑模式，在文件末尾添加如下内容。
   ```
   [ceph-spdk]
   name=ceph-ksal
   baseurl=file:///home/ceph-ksal
   enabled=1
   gpgcheck=0
   priority=1
   ```
6. 按“Esc”键退出编辑模式，输入**:wq!**并按“Enter”键保存退出文件。

步骤3 在所有节点将pip源配置为华为镜像源，提高下载速度。

1. 新建“.pip”路径并在路径下创建“pip.conf”文件。
   ```
   mkdir -p ~/.pip
   vi ~/.pip/pip.conf
   ```
2. 按“i”进入编辑模式，添加如下内容。
   ```
   [global]
   timeout = 120
   index-url = https://repo.huaweicloud.com/repository/pypi/simple
   trusted-host = repo.huaweicloud.com
   ```
3. 按“Esc”键退出编辑模式，输入**:wq!**并按“Enter”键保存退出文件。

----结束

关闭防火墙

须知

Linux系统默认开启的防火墙安全机制会阻止各组件之间的正常连接，导致无法正常部署Ceph集群。这是Linux OS本身的行为，鲲鹏BoostKit分布式存储KSAL并不对此提供的详细解决方案。如果客户想在自己的系统中开启防火墙，则需要自行寻找解决方法。

针对此限制，本文提供了快速的关闭防火墙的方法。鲲鹏BoostKit分布式存储KSAL特性中，Host OS不在交付范围内，提供的防火墙配置方法仅供参考，不是商用交付的一部分，不做商用承诺。如果商用，需自行评估并承担相应风险。

关闭防火墙可能会导致安全问题，如果客户最终的解决方案本身就没有规划启用防火墙，建议通过端到端的整体方案来弥补防火墙禁用带来的风险，且需自行承担安全风险。如果客户有开启防火墙的需求，建议对防火墙进行细粒度的规则配置，根据实际使用情况，开放对应的端口、协议，并保证整个系统的安全。

以下命令仅关闭当前节点防火墙，需在所有节点依次执行。

```
systemctl stop firewalld.service
systemctl disable firewalld.service
systemctl status firewalld.service
```

配置主机名

配置永久静态主机名，建议将服务端节点配置为ceph1~ceph3，客户端节点配置为client1~client3。

步骤1 配置节点名称。

1. 为ceph1节点配置。
   ```
   hostnamectl --static set-hostname ceph1
   ```
2. 参照**步骤1.1**依次为ceph2和ceph3节点配置。
3. 为client1节点配置。
   ```
   hostnamectl --static set-hostname client1
   ```
4. 参照**步骤1.3**依次为client2和client3节点配置。

步骤2 在所有节点修改域名解析文件。

1. 打开文件。
   ```
   vi /etc/hosts
   ```
2. 按“i”键进入编辑模式，在文件中加入以下内容。
   ```
   192.168.3.166  ceph1
   192.168.3.167  ceph2
   192.168.3.168  ceph3
   192.168.3.160  client1
   192.168.3.161  client2
   192.168.3.162  client3
   ```

   说明

   - IP地址为**集群规划**中所规划的IP地址，仅为示例，请根据实际情况进行修改。可通过**ip a**命令查询实际IP地址。
   - 本文规划集群为三台服务端节点和三台客户端节点，请根据实际节点数量对文件内容进行调整。

3. 按“Esc”键退出编辑模式，输入**:wq!**并按“Enter”键保存退出文件。

----结束

配置 NTP

Ceph中会自动校验集群节点之间的时间，若不同节点之间时差较大，会产生告警，因此需要在各节点间配置时钟同步。

步骤1 在所有节点安装NTP服务。

```
yum -y install ntp ntpdate
```

步骤2 在所有节点备份旧配置。

```
cd /etc && mv ntp.conf ntp.conf.bak
```

步骤3 将ceph1配置为NTP服务端节点。

1. 在ceph1新建NTP配置文件。
   ```
   vi /etc/ntp.conf
   ```
2. 按“i”键进入编辑模式，新增如下内容。
   ```
   restrict 127.0.0.1
   restrict ::1
   restrict 192.168.3.0 mask 255.255.255.0
   server 127.127.1.0
   fudge 127.127.1.0 stratum 8
   ```

   说明

   以上文件中，“restrict 192.168.3.0 mask 255.255.255.0”是ceph1所在的Public网段IP地址与掩码。
3. 按“Esc”键退出编辑模式，输入**:wq!**并按“Enter”键保存退出文件。

步骤4 将除ceph1以外的其他所有节点配置为NTP客户端节点。

1. 在除ceph1以外的其他所有节点新建NTP文件。
   ```
   vi /etc/ntp.conf
   ```
2. 按“i”键进入编辑模式，新增如下内容。此处输入的IP地址为ceph1的IP地址。
   ```
   server 192.168.3.166
   ```
3. 按“Esc”键退出编辑模式，输入**:wq!**并按“Enter”键保存退出文件。

步骤5 启动NTP服务。

1. 在ceph1节点启动NTP服务，并检查状态。
   ```
   systemctl start ntpd.service
   systemctl enable ntpd.service
   systemctl status ntpd.service
   ```
   回显如下所示。
   ```
   [root@ceph1 ~]# systemctl start ntpd.service
   [root@ceph1 ~]# systemctl enable ntpd.service
   [root@ceph1 ~]# systemctl status ntpd.service
   ● ntpd.service - Network Time Service
     Loaded: loaded (/usr/lib/systemd/system/ntpd.service; enabled; vendor preset: disabled)
     Active: active (running) since Fri 2023-09-01 10:14:56 CST; 8s ago
   Main PID: 129667 (ntpd)
      Tasks: 2
     Memory: 3.5M
     CGroup: /system.slice/ntpd.service
             └─129667 /usr/sbin/ntpd -u ntp:ntp -g

   Sep 01 10:14:56 ceph1 ntpd[129667]: Listen normally on 4 bond2 192.168.2.128:123
   Sep 01 10:14:56 ceph1 ntpd[129667]: Listen normally on 5 bond1 192.168.1.128:123
   Sep 01 10:14:56 ceph1 ntpd[129667]: Listen normally on 6 lo [::1]:123
   ```



```
Sep 01 10:14:56 ceph1 ntpd[129667]: Listen normally on 7 enp125s0f0 [fe80::5d5a:faa4:75a4:4afa
%2]:123
Sep 01 10:14:56 ceph1 ntpd[129667]: Listen normally on 8 bond2 [fe80::a975:9916:4607:a50b
%12]:123
Sep 01 10:14:56 ceph1 ntpd[129667]: Listen normally on 9 bond1 [fe80::147b:9453:cfc0:d19f%13]:123
Sep 01 10:14:56 ceph1 ntpd[129667]: Listening on routing socket on fd #26 for interface updates
Sep 01 10:14:56 ceph1 ntpd[129667]: kernel reports TIME_ERROR: 0x2041: Clock Unsynchronized
Sep 01 10:14:56 ceph1 ntpd[129667]: kernel reports TIME_ERROR: 0x2041: Clock Unsynchronized
Sep 01 10:14:56 ceph1 systemd[1]: Started Network Time Service.
```

2. NTP服务启动后等待5分钟，在除ceph1的所有节点强制同步NTP服务端（ceph1）时间。
   ```
   ntpdate ceph1
   ```
3. 在除ceph1的所有节点写入硬件时钟，避免重启后失效。
   ```
   hwclock -w
   ```
4. 在除ceph1的所有节点安装并启动crontab工具。
   ```
   yum install -y crontabs
   systemctl enable crond.service
   systemctl start crond.service
   crontab -e
   ```
5. 按“i”键进入编辑模式，添加以下内容，每隔10分钟自动与ceph1同步时间。
   ```
   */10 * * * * /usr/sbin/ntpdate 192.168.3.166
   ```
6. 按“Esc”键退出编辑模式，输入**:wq!**并按“Enter”键保存退出文件。

----结束

配置免密登录

步骤1 在ceph1节点生成公钥。

```
ssh-keygen -t rsa
```

按“Enter”键使用默认配置。

步骤2 在ceph1节点将公钥发放至集群其他节点。

```
for i in {1..3};do ssh-copy-id ceph$i;done
for i in {1..3};do ssh-copy-id client$i;done
```

根据提示确认操作并输入节点root密码。

----结束

关闭 SELinux

须知

Linux系统默认开启的SELinux安全机制会阻止各组件之间的正常连接，导致无法正常部署Ceph集群。这是Linux OS本身的行为，鲲鹏BoostKit分布式存储使能套件并不对此提供的详细解决方案。如果客户想在自己的系统中使用SELinux，则需要自行寻找解决方法。

针对此限制，我们提供了快速的禁用SELinux的方法。鲲鹏BoostKit分布式存储使能套件中，提供的SELinux配置方法仅供参考，需客户自行评估并承担相应风险。

禁用SELinux可能会导致安全问题，如果客户最终的解决方案本身就没有规划启用SELinux，建议通过端到端的整体方案来弥补SELinux关闭带来的风险，且需自行承担安全风险。如果客户有SELinux的需求，建议根据实际的SELinux问题进行细粒度的规则配置，并保证整个系统的安全。

在所有节点关闭SELinux。

- 方式一：临时关闭，重启服务器后失效。
  ```
  setenforce permissive
  ```
- 方式二：永久关闭，下一次重启服务器自动生效。
  a. 打开config文件。
  ```
  vi /etc/selinux/config
  ```
  b. 按“i”键进入编辑模式，修改SELINUX=**permissive**。
  ```
  [root@ceph1 ~]# cat /etc/selinux/config

  # This file controls the state of SELinux on the system.
  # SELINUX= can take one of these three values:
  #     enforcing - SELinux security policy is enforced.
  #     permissive - SELinux prints warnings instead of enforcing.
  #     disabled - No SELinux policy is loaded.
  SELINUX=permissive
  # SELINUXTYPE= can take one of these three values:
  #     targeted - Targeted processes are protected,
  #     minimum - Modification of targeted policy. Only selected processes are protected.
  #     mls - Multi Level Security protection.
  SELINUXTYPE=targeted
  ```
  c. 按“Esc”键退出编辑模式，输入**:wq!**并按“Enter”键保存退出文件。

6.3.2 安装 Ceph 软件

在所有节点安装Ceph软件，并在ceph1节点安装ceph-deploy工具方便后续集群部署。

步骤1 设置所有节点Yum证书验证状态为不验证。

1. 打开“yum.conf”文件。
   ```
   vi /etc/yum.conf
   ```
2. 按“i”键进入编辑模式，添加如下内容至文件末尾。
   ```
   sslverify=false
   deltarpm=0
   ```
3. 按“Esc”键退出编辑模式，输入**:wq!**并按“Enter”键保存退出文件。

步骤2 在所有节点安装gperftools工具。

说明

openEuler LTS SP的Yum源中gperftools-libs 2.8.1版本用的是gperftools-libs 2.8.0的源码，存在bug，因此不能使用该版本RPM包。

建议下载openEuler LTS的Yum源下gperftools-devel-2.7.7、gperftools-libs-2.7.7的RPM包，配置成本地源。

1. 下载gperftools-devel-2.7.7、gperftools-libs-2.7.7的RPM包。
   ```
   mkdir -p /home/gperftools-2.7-7 && cd /home/gperftools-2.7-7
   wget --no-check-certificate  https://repo.openeuler.org/openEuler-20.03-LTS/OS/aarch64/Packages/gperftools-devel-2.7-7.oe1.aarch64.rpm
   wget --no-check-certificate  https://repo.openeuler.org/openEuler-20.03-LTS/OS/aarch64/Packages/gperftools-libs-2.7-7.oe1.aarch64.rpm
   createrepo .
   ```
2. 打开“/etc/yum.repos.d/openEuler.repo”文件。
   ```
   vi /etc/yum.repos.d/openEuler.repo
   ```
3. 按“i”键进入编辑模式，添加如下内容至文件末尾。
   ```
   [gperftools-2.7-7]
   name=gperftools-2.7-7
   baseurl=file:///home/gperftools-2.7-7
   enabled=1
   gpgcheck=0
   priority=1
   ```


4. 按“Esc”键退出编辑模式，输入**:wq!**并按“Enter”键保存退出文件。

步骤3 在所有节点安装Ceph。

```
dnf -y install librados2-14.2.8 ceph-14.2.8
pip install prettytable werkzeug
```

说明

若安装失败，请参见**步骤3**配置pip华为镜像源，提升下载速度。

步骤4 在ceph1节点额外安装ceph-deploy。

```
pip install ceph-deploy
```

步骤5 配置ceph-deploy适配openEuler操作系统。

1. 在ceph1节点打开ceph_deploy的“__init__.py”文件。
   - openEuler 20.03：
     ```
     vi /lib/python2.7/site-packages/ceph_deploy/hosts/__init__.py
     ```
   - openEuler 22.03：
     ```
     vi /lib/python3.9/site-packages/ceph_deploy/hosts/__init__.py
     ```

   说明

   OpenEuler 20.03默认的python版本是2.7，OpenEuler 22.03默认的python是3.9。
2. 按“i”键进入编辑模式，在“_get_distro”函数中增加如下代码。
   ```
   'openeuler':fedora,
   ```
3. 按“Esc”键退出编辑模式，输入**:wq!**并按“Enter”键保存退出文件。

步骤6 在各节点查看版本。

```
ceph -v
```

查询结果如下所示。

```
ceph version 14.2.8 (2d095e947a02261ce61424021bb43bd3022d35cb) nautilus (stable)
```

----结束

6.3.3 部署 MON 节点

在ceph1节点执行部署MON（Monitor）节点的所有操作。

步骤1 创建集群。

```
cd /etc/ceph
ceph-deploy new ceph1 ceph2 ceph3
```

回显如下所示。

```
[ceph_deploy.conf][DEBUG ] found configuration file at: /root/.cephdeploy.conf
[ceph_deploy.cli][INFO  ] Invoked (2.0.1): /home/xiaoshuang/0808/kunpeng_perfstudiokit/venv/bin/ceph-deploy new ceph1 ceph2 ceph3
[ceph_deploy.cli][INFO  ] ceph-deploy options:
[ceph_deploy.cli][INFO  ]  verbose                       : False
[ceph_deploy.cli][INFO  ]  quiet                         : False
[ceph_deploy.cli][INFO  ]  username                      : None
[ceph_deploy.cli][INFO  ]  overwrite_conf                : False
[ceph_deploy.cli][INFO  ]  ceph_conf                     : None
[ceph_deploy.cli][INFO  ]  cluster                       : ceph
[ceph_deploy.cli][INFO  ]  mon                           : ['ceph1', 'ceph2', 'ceph3']
[ceph_deploy.cli][INFO  ]  ssh_copykey                   : True
[ceph_deploy.cli][INFO  ]  fsid                          : None
[ceph_deploy.cli][INFO  ]  cluster_network               : None
[ceph_deploy.cli][INFO  ]  public_network                : None
[ceph_deploy.cli][INFO  ]  cd_conf                       : <ceph_deploy.conf.cephdeploy.Conf object at
```

```
0xfffd10452890>
[ceph_deploy.cli][INFO  ]  default_release               : False
[ceph_deploy.cli][INFO  ]  func                          : <function new at 0xfffd1044e830>
[ceph_deploy.new][DEBUG ] Creating new cluster named ceph
[ceph_deploy.new][INFO  ] making sure passwordless SSH succeeds
[ceph1][DEBUG ] connected to host: ceph1
```

步骤2 修改相关配置。

修改用于将内部集群间的网络与外部访问的网络隔离。配置192.168.4.0用于内部存储集群之间的数据同步（仅在服务端节点间使用），192.168.3.0用于服务端节点与客户端节点的数据交互。

1. 打开“/etc/ceph”目录下自动生成的“ceph.conf”文件。

   说明

   配置节点命令以及使用ceph-deploy配置OSD时，需在“/etc/ceph”目录下执行，否则会报错。

   ```
   vi /etc/ceph/ceph.conf
   ```

2. 按“i”键进入编辑模式，对文件内容进行如下修改。

   ```
   [global]
   fsid = f5a4f55c-d25b-4339-a1ab-0fceb4a2996f
   mon_initial_members = ceph1, ceph2, ceph3
   mon_host = 192.168.3.166,192.168.3.167,192.168.3.168
   auth_cluster_required = cephx
   auth_service_required = cephx
   auth_client_required = cephx

   public_network = 192.168.3.0/24
   cluster_network = 192.168.4.0/24

   [mon]
   mon_allow_pool_delete = true
   ```

   须知

   Ceph 14.2.8版本在使用bluestore引擎的时候默认会打开bluefs的buffer开关，可能导致系统下内存全部被buff/cache占用，使性能下降。可以采用以下两种方案解决：

   - 在集群压力不大的场景下可以将“bluefs_buffered_io”开关设置成“false”。
   - 可以通过定时执行**echo 3 > /proc/sys/vm/drop_caches**来强制回收“buffer/cache”中的内存。

3. 按“Esc”键退出编辑模式，输入**:wq!**并按“Enter”键保存退出文件。

步骤3 初始化监视器并收集密钥。

```
ceph-deploy mon create-initial
```

执行完成后，脚本会自动生成ceph.client.admin.keyring，回显如下所示。

```
[ceph_deploy.conf][DEBUG ] found configuration file at: /root/.cephdeploy.conf
[ceph_deploy.cli][INFO  ] Invoked (2.0.1): /usr/bin/ceph-deploy mon create-initial
[ceph_deploy.cli][INFO  ] ceph-deploy options:
[ceph_deploy.cli][INFO  ]  username                      : None
[ceph_deploy.cli][INFO  ]  verbose                       : False
[ceph_deploy.cli][INFO  ]  overwrite_conf                : False
[ceph_deploy.cli][INFO  ]  subcommand                    : create-initial
[ceph_deploy.cli][INFO  ]  quiet                         : False
[ceph_deploy.cli][INFO  ]  cd_conf                       : <ceph_deploy.conf.cephdeploy.Conf instance at
```

```
0xfffcbf407f00>
[ceph_deploy.cli][INFO  ]  cluster                : ceph
[ceph_deploy.cli][INFO  ]  func                   : <function mon at 0xfffcbf3e75d0>
[ceph_deploy.cli][INFO  ]  ceph_conf              : None
[ceph_deploy.cli][INFO  ]  keyrings               : None
[ceph_deploy.cli][INFO  ]  default_release         : False
[ceph_deploy.mon][DEBUG ] Deploying mon, cluster ceph hosts ceph1 ceph2 ceph3
[ceph_deploy.mon][DEBUG ] detecting platform for host ceph1 ...
[ceph1][DEBUG ] connected to host: ceph1
[ceph1][DEBUG ] detect platform information from remote host
[ceph1][DEBUG ] detect machine type
[ceph1][DEBUG ] find the location of an executable
[ceph_deploy.mon][INFO  ] distro info: openEuler 20.03 LTS-SP1
[ceph1][DEBUG ] determining if provided host has same hostname in remote
[ceph1][DEBUG ] get remote short hostname
[ceph1][DEBUG ] deploying mon to ceph1
[ceph1][DEBUG ] get remote short hostname
[ceph1][DEBUG ] remote hostname: ceph1
[ceph1][DEBUG ] write cluster configuration to /etc/ceph/{cluster}.conf
[ceph1][DEBUG ] create the mon path if it does not exist
```

步骤4 将ceph.client.admin.keyring拷贝到各个节点上。

```
ceph-deploy --overwrite-conf admin ceph1 ceph2 ceph3 client1 client2 client3
```

执行成功回显如下所示。

```
[ceph_deploy.conf][DEBUG ] found configuration file at: /root/.cephdeploy.conf
[ceph_deploy.cli][INFO  ] Invoked (2.0.1): /usr/bin/ceph-deploy --overwrite-conf admin ceph1 ceph2 ceph3
client1
[ceph_deploy.cli][INFO  ] ceph-deploy options:
[ceph_deploy.cli][INFO  ]  username                : None
[ceph_deploy.cli][INFO  ]  verbose                 : False
[ceph_deploy.cli][INFO  ]  overwrite_conf          : True
[ceph_deploy.cli][INFO  ]  quiet                   : False
[ceph_deploy.cli][INFO  ]  cd_conf                 : <ceph_deploy.conf.cephdeploy.Conf instance at
0xfffd01a83f50>
[ceph_deploy.cli][INFO  ]  cluster                 : ceph
[ceph_deploy.cli][INFO  ]  client                  : ['ceph1', 'ceph2', 'ceph3', 'client1']
[ceph_deploy.cli][INFO  ]  func                    : <function admin at 0xfffd01c495d0>
[ceph_deploy.cli][INFO  ]  ceph_conf               : None
[ceph_deploy.cli][INFO  ]  default_release          : False
[ceph_deploy.admin][DEBUG ] Pushing admin keys and conf to ceph1
[ceph1][DEBUG ] connected to host: ceph1
[ceph1][DEBUG ] detect platform information from remote host
[ceph1][DEBUG ] detect machine type
[ceph1][DEBUG ] write cluster configuration to /etc/ceph/{cluster}.conf
[ceph_deploy.admin][DEBUG ] Pushing admin keys and conf to ceph2
[ceph2][DEBUG ] connected to host: ceph2
```

步骤5 查看是否配置成功。

```
ceph -s
```

配置成功回显如下所示。

```
cluster:
id:     f6b3c38c-7241-44b3-b433-52e276dd53c6
health: HEALTH_OK
services:
mon: 3 daemons, quorum ceph1,ceph2,ceph3 (age 25h)
```

----结束

6.3.4 部署 MGR 节点

在ceph1节点执行部署MGR（Manager）节点的所有操作。

步骤1 部署MGR节点。


```
ceph-deploy mgr create ceph1 ceph2 ceph3
```

执行成功回显如下所示。

```
[ceph_deploy.conf][DEBUG ] found configuration file at: /root/.cephdeploy.conf
[ceph_deploy.cli][INFO  ] Invoked (2.0.1): /usr/bin/ceph-deploy mgr create ceph1 ceph2 ceph3
[ceph_deploy.cli][INFO  ] ceph-deploy options:
[ceph_deploy.cli][INFO  ]  username                      : None
[ceph_deploy.cli][INFO  ]  verbose                       : False
[ceph_deploy.cli][INFO  ]  mgr                           : [('ceph1', 'ceph1'), ('ceph2', 'ceph2'), ('ceph3', 'ceph3')]
[ceph_deploy.cli][INFO  ]  overwrite_conf                : False
[ceph_deploy.cli][INFO  ]  subcommand                    : create
[ceph_deploy.cli][INFO  ]  quiet                         : False
[ceph_deploy.cli][INFO  ]  cd_conf                       : <ceph_deploy.conf.cephdeploy.Conf instance at
0xffff41bc7280>
[ceph_deploy.cli][INFO  ]  cluster                       : ceph
[ceph_deploy.cli][INFO  ]  func                          : <function mgr at 0xffff41c57bd0>
[ceph_deploy.cli][INFO  ]  ceph_conf                     : None
[ceph_deploy.cli][INFO  ]  default_release               : False
[ceph_deploy.mgr][DEBUG ] Deploying mgr, cluster ceph hosts ceph1:ceph1 ceph2:ceph2 ceph3:ceph3
[ceph1][DEBUG ] connected to host: ceph1
[ceph1][DEBUG ] detect platform information from remote host
[ceph1][DEBUG ] detect machine type
[ceph_deploy.mgr][INFO  ] Distro info: openEuler 20.03 LTS-SP1
[ceph_deploy.mgr][DEBUG ] remote host will use systemd
[ceph_deploy.mgr][DEBUG ] deploying mgr bootstrap to ceph1
[ceph1][DEBUG ] write cluster configuration to /etc/ceph/{cluster}.conf
```

步骤2 查看MGR是否部署成功。

```
ceph -s
```

配置成功回显如下所示。

```
cluster:
id:     f6b3c38c-7241-44b3-b433-52e276dd53c6
health: HEALTH_OK

services:
mon: 3 daemons, quorum ceph1,ceph2,ceph3 (age 25h)
mgr: ceph1(active, since 2d), standbys: ceph2, ceph3
```

----结束

6.3.5 部署 OSD 节点

根据具体的操作节点提示完成部署OSD（Object Storage Device）的所有操作。

划分 OSD 分区

须知

以下操作均需在ceph1~ceph3均执行，此处以“/dev/nvme0n1”为例说明，如果有多块NVMe SSD或SATA/SAS接口SSD，只需将脚本中的“/dev/nvme0n1”盘符替换为对应盘符即可。

对于非推荐配置，若NVMe盘划分的DB分区和WAL分区空间不足，将会放到HDD中，影响性能。

如下操作将NVMe盘划分为12个60GB分区、12个180GB分区，分别对应WAL分区、DB分区。

步骤1 创建分区脚本。

1. 新建“partition.sh”文件。
   vi partition.sh
2. 按“i”键进入编辑模式，添加如下内容。

```
#!/bin/bash

parted /dev/nvme0n1 mklabel gpt

for j in `seq 1 12`
do
((b = $(( $j * 8 ))))
((a = $(( $b - 8 ))))
((c = $(( $b - 6 ))))
str="%"
echo $a
echo $b
echo $c
parted /dev/nvme0n1 mkpart primary ${a}${str} ${c}${str}
parted /dev/nvme0n1 mkpart primary ${c}${str} ${b}${str}
done
```

   说明

   此脚本内容只适用于当前硬件配置，其它硬件配置可参考此脚本。

3. 按“Esc”键退出编辑模式，输入**:wq!**并按“Enter”键保存退出文件。

步骤2 执行脚本，创建分区。

```
bash partition.sh
```

----结束

部署 OSD 节点

说明

以下脚本的“/dev/sda-/dev/sdl”12块硬盘均为数据盘，OS安装在“/dev/sdm”上。实际情况中可能会遇到系统盘位于数据盘中间的情况，比方说系统盘安装到了“/dev/sde”，则不能直接使用以下脚本直接运行，否则部署到“/dev/sde”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系统盘、做DB/WAL分区的SSD盘等。

步骤1 确认各个节点各硬盘的sd*。

```
lsblk
```

回显如下所示，/dev/sda是系统盘。

```
[root@client1 ~]# lsblk
NAME                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda                   8:0    0  3.7T  0 disk
├─sda1                8:1    0    1G  0 part /boot/efi
├─sda2                8:2    0    1G  0 part /boot
└─sda3                8:3    0  3.7T  0 part
  ├─openeuler-root 253:0    0  3.6T  0 lvm  /
  └─openeuler-swap 253:1    0    4G  0 lvm  [SWAP]
```

说明

有一些硬盘可能是以前Ceph集群里的数据盘或者曾经安装过操作系统，那么这些硬盘上很可能有未清理的分区，**lsblk**命令可以看到各个硬盘下是否有分区。假如/dev/sdb硬盘下发现有分区信息，可用如下命令清除：

```
ceph-volume lvm zap /dev/sdb --destroy
```

必须先确定哪些盘做为数据盘使用，当数据盘有未清理的分区时再执行清除命令。

步骤2 在ceph1节点创建OSD部署脚本。

1. 新建“create_osd.sh”文件。
   ```
   cd /etc/ceph/
   vi /etc/ceph/create_osd.sh
   ```
2. 按“i”键进入编辑模式，添加如下内容。
   ```
   #!/bin/bash

   for node in ceph1 ceph2 ceph3
   do
   j=1
   k=2
   for i in {a..l}
   do
   ceph-deploy osd create ${node} --data /dev/sd${i} --block-wal /dev/nvme0n1p${j} --block-db /dev/
   nvme0n1p${k}
   ((j=${j}+2))
   ((k=${k}+2))
   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。

   其中，**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。
   - --block-wal选项后面是WAL分区。
   - --block-db选项后面是DB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--block-db和--block-wal，只需要--data指定数据盘即可。
3. 按“Esc”键退出编辑模式，输入**:wq!**并按“Enter”键保存退出文件。

步骤3 在ceph1上运行脚本。

```
bash create_osd.sh
```

步骤4 创建成功后，查看是否正常，即36个OSD是否都为up。

```
ceph -s
```

----结束

6.4 卸载 Ceph

若不再需要使用Ceph服务，可卸载Ceph。

步骤1 停止所有Ceph服务进程。

```
systemctl stop ceph.target
```

步骤2 卸载Ceph。

```
yum rm ceph
```

步骤3 卸载Ceph相关目录。

```
rm -rf /var/lib/ceph/*
rm -rf /etc/ceph/*
rm -rf /var/run/ceph/*
```

----结束

6.5 使能 KSAL EC 算法

创建EC存储池使能KSAL EC特性。

请参见《**Ceph对象存储 部署指南（CentOS 7.6&openEuler 20.03）**》中的“创建EC存储池”相关内容，完成EC存储池的创建。

创建EC profile时须启用KSAL插件，即增加“plugin=ksal”参数。因此在创建EC profile时请使用如下命令替换原文命令。

```
ceph osd erasure-code-profile set myprofile k=4 m=2 crush-failure-domain=osd crush-device-class=hdd plugin=ksal
```

说明

当集群的节点数量大于等于6（k+m）时，“crush-failure-domain”可以配置成host。

7 KSAL 开发参考

7.1 CRC32C 接口

计算字符数组的CRC32C校验和。

接口格式

uint32_t KsalCrc32c(uint32_t seed, uint8_t *data, uint64_t len);

参数释义

参数名	类型	描述	输入/输出
seed	32位整型数	CRC的种子。	输入
data	字符数组	CRC的字符数组。	输入
len	64位整型数	数组长度。	输入
crc	32位整型数	CRC校验和。	输出

使用依赖

ksal/ksal_crc.h

使用实例

步骤1 编写CRC32C测试代码。

1. 新建“test.c”文件。
   ```
   vi test.c
   ```
2. 按“i”键进入编辑模式，添加如下测试代码。
   ```
   #include <stdio.h>
   #include <stdlib.h>
   #include <ksal/ksal_crc.h>

   int main(int argc, char **argv)
   {
       uint8_t buf[4096];
       for (size_t i = 0; i < sizeof(buf); ++i) {
           buf[i] = (int8_t)i & 0xff;
       }
       uint32_t crc = KsalCrc32c(0x123, buf, sizeof(buf));
       printf("crc = 0x%x\r\n", crc);
       return 0;
   }
   ```
3. 按“Esc”键退出编辑模式，输入**:wq!**并按“Enter”键保存退出文件。

步骤2 编译“test.c”文件，生成test可执行文件。

```
gcc test.c -lksal -o test
```

步骤3 运行test可执行文件。

```
./test
```

执行输出如下。

```
crc = 0x4ee4ecc1
```

----结束

7.2 CRC16 T10DIF 接口

计算字符数组的CRC16 T10DIF校验和。

接口格式

uint16_t KsalCrc16T10Dif(uint16_t seed, uint8_t *data, uint64_t len);

参数释义

参数名	类型	描述	输入/输出
seed	16位整型数	CRC的种子。	输入
data	字符数组	CRC的字符数组。	输入
len	64位整型数	数组长度。	输入
crc	16位整型数	CRC校验和。	输出

使用依赖

ksal/ksal_crc.h

使用实例

步骤1 编写CRC16 T10DIF测试代码。

1. 新建“test.c”文件。
   ```
   vi test.c
   ```
2. 按“i”键进入编辑模式，添加如下测试代码。
   ```
   #include <stdio.h>
   #include <stdlib.h>
   #include <ksal/ksal_crc.h>

   int main(int argc, char **argv)
   {
       uint8_t buf[4096];
       for (size_t i = 0; i < sizeof(buf); ++i) {
           buf[i] = (int8_t)i & 0xff;
       }
       uint16_t crc = KsalCrc16T10Dif(0x123, buf, sizeof(buf));
       printf("crc = 0x%x\r\n", crc);
       return 0;
   }
   ```
3. 按“Esc”键退出编辑模式，输入**:wq!**并按“Enter”键保存退出文件。

步骤2 编译“test.c”文件，生成test可执行文件。

```
gcc test.c -lksal -o test
```

步骤3 运行test可执行文件。

```
./test
```

执行输出如下。

```
crc = 0xec6a
```

----结束

7.3 EC 接口

介绍EC编解码接口。

接口格式

int KsalEcEncode(uint8_t **data, uint8_t **parity, struct HecParam *param);

int KsalEcDecode(uint8_t **data, uint8_t **parity, struct HecParam *param);

参数释义

参数名	类型	描述	输入/输出
data	指针数组	数据块指针数组。	输入/输出
parity	指针数组	校验块指针数组。	输入/输出
param	编解码参数指针	编解码参数。	输入
ret	整型数	返回0表示成功， 其他则表示失败。	输出

数据结构

成员名	类型	描述
dataNum	整型数	数据块个数。
parityNum	整型数	校验块个数。
sectorSize	整型数	扇区大小，最小编解码粒度，当前支持4096和4160。
blockSize	整型数	数据块或者校验块的大小，须为sectorSize的整数倍。
targetColNum	整型数	损失块的个数，解码接口使用。
targetColArray[6]	整型数	损失块列表，各个元素取值范围为[0, dataNum + parityNum)。
version	整型数	算法版本号，当前为0。

使用依赖

ksal/ksal_erasure_code.h

使用实例

步骤1 编写EC测试代码。

1. 新建“test.c”文件。
   ```
   vi test.c
   ```
2. 按“i”键进入编辑模式，添加如下测试代码。
   ```
   #include <stdio.h>
   #include <stdlib.h>
   #include <string.h>
   #include <ksal/ksal_erasure_code.h>

   int main(int argc, char **argv)
   {
       struct HecParam param = {4, 2, 4096, 8192, 2, {3, 4}};
       uint8_t tmp[20][8192];
       uint8_t *ptrs[20];
       uint8_t *bptrs[20];
       for (size_t i = 0; i < param.dataNum + param.parityNum; ++i) {
           for (size_t j = 0; j < param.blockSize; j++) {
                   tmp[i][j] = (uint8_t)rand() & 0xff;
           }
           ptrs[i] = tmp[i];
           bptrs[i] = tmp[i];
       }
       KsalEcEncode(ptrs, ptrs + param.dataNum, &param);
       bptrs[3] = tmp[18];
       bptrs[4] = tmp[19];
       KsalEcDecode(bptrs, bptrs + param.dataNum, &param);
       if (memcmp(bptrs[3], ptrs[3], param.blockSize) == 0 &&
           memcmp(bptrs[4], ptrs[4], param.blockSize) == 0) {
           printf("decode succ!!\r\n");
       } else {
           printf("decode fail!!\r\n");
       }
   ```



```
    return 0;
}
```

3. 按“Esc”键退出编辑模式，输入**:wq!**并按“Enter”键保存退出文件。

步骤2 编译“test.c”文件，生成test可执行文件。

```
gcc test.c -lksal -o test
```

步骤3 运行test可执行文件。

```
./test
```

执行输出如下。

```
decode succ!!
```

----结束

8 缩略语

缩略语	英文全称	中文全称
A-E		
CRC	Cyclic Redundancy Check	循环冗余码校验
DIF	Data Integrity Field	数据完整性字段
EC	Erasure Code	纠删码
F-J		
GIV	Global Industry Vision	全球产业展望
K-O		
KSAL	Kunpeng Storage Acceleration Library	鲲鹏存储加速算法库
MON	Monitor	监控节点
MGR	Manager	管理节点
NTP	Network Time Protocol	网络时间协议
OSD	Object Storage Device	对象存储设备
X		
XOR	Exclusive OR	异或

9 版本配套信息

9.1 产品版本信息

产品名称	Kunpeng BoostKit
产品版本	23.0.0
软件名称	KSAL（Kunpeng Storage Acceleration Library）
软件版本	1.1.0

9.2 软件版本配套说明

软件名称	软件版本
OS	openEuler 20.03 LTS SP1 openEuler 22.03 LTS SP1
Ceph	14.2.8

9.3 硬件版本配套说明

项目	要求
CPU型号	鲲鹏920处理器

项目	要求
服务器型号	TaiShan 200服务器（型号2280） TaiShan 200服务器（型号5280）

9.4 病毒扫描结果

本软件包经过OfficeScan、Bitdefender、Kaspersky防病毒软件扫描，没有发现病毒。详细信息如下：

Engine Name	OfficeScan
Engine Version	12.000.1008
Virus Lib Version	1877160
Scan Time	2023-01-03 14:15:45
Scan Result	OK

Engine Name	Bitdefender
Engine Version	7.0.3.2050
Virus Lib Version	7.95471
Scan Time	2023-01-03 14:15:50
Scan Result	OK

Engine Name	Kaspersky
Engine Version	11.2.0.4528
Virus Lib Version	2023-01-03 00:58:00
Scan Time	2023-01-03 14:15:59
Scan Result	OK

10 1.1.0

10.1 更新说明

新增特性

表 10-1 1.1.0 版本新增特性

特性名称	更新说明
EC算法	新增支持条带为64Byte粒度的编解码

10.2 已解决的问题

无。

10.3 遗留问题

无。

11 1.0.0

11.1 更新说明

本次发布为第一次正式发布。

KSAL是华为自研的存储算法加速库，当前包括EC算法、CRC16 T10DIF算法和CRC32C算法。

新增特性

表 11-1 1.0.0 版本新增特性

特性名称	更新说明
EC算法	基于华为自研向量化EC（Erasure Code）编解码方案，通过同构映射将EC编码过程中所需的高阶有限域GF(2^w)乘法操作替换为二元矩阵乘法，进而将查表实现的复杂有限域乘法操作替代为XOR（Exclusive OR）操作，同时采用编码编排算法在校验块计算过程中对中间结果进行复用，减少XOR操作数，配合鲲鹏向量化指令实现编码加速。与主流开源EC算法相比，性能提升1倍以上。
CRC16 T10DIF算法	CRC16 T10DIF（Cyclic Redundancy Check 16 T10 Data Integrity Field）算法和通过大数求余算法和配合鲲鹏向量化指令实现编码加速，相比开源算法，性能提升2倍以上。
CRC32C算法	CRC32C（Cyclic Redundancy Check 32 Castagnoli）算法通过大数求余算法和配合鲲鹏向量化指令实现编码加速，相比开源算法，性能提升20%以上。

11.2 已解决的问题

无。

11.3 遗留问题

无。

12 版本配套文档

12.1 1.1.0 版本配套文档

文档名称	内容简介	交付形式
《Kunpeng BoostKit 23.0.0 分布式存储加速算法库 版本说明书》	本文档提供加速算法版本发布及其配套信息。	华为企业业务网站 鲲鹏社区
《Kunpeng BoostKit 23.0.0 分布式存储加速算法库 开发指南》	本文档提供加速算法部署使能及开发指导。	

12.2 获取文档的方法

您可以通过以下方式获取文档：

- 方式1：访问**华为企业业务网站**浏览和获取相关的文档。

  说明

  在网站上获取文档，需要申请相应的用户权限。如果您是初次登录网站，需要先注册。详细的操作方法请参考网站帮助和FAQ。

- 方式2：访问**鲲鹏社区**浏览获取相关文档。

13 漏洞修补列表

无。

14 缩略语

缩略语	英文全称	中文全称
CRC	Cyclic Redundancy Check	循环冗余码校验
DIF	Data Integrity Field	数据完整性字段
EC	Erasure Code	纠删码
KSAL	Kunpeng Storage Acceleration Library	鲲鹏存储加速算法库
XOR	Exclusive OR	异或

15 修订记录

发布日期	修改说明
2024-01-11	第一次正式发布。