目录

1. 实验一：优化 Linux 系统的内存

1.1. 实验目的

完成本实验，您应该能够：

- 掌握给用户添加 root 组权限的方法
- 掌握优化交换内存（swap）的方法
- 掌握优化内存分配策略的方法
- 掌握优化脏页配置的方法

1.2. 实验要求

- 熟悉常用 Linux 操作系统命令
- 理解 Linux 用户及权限管理
- 了解交换内存（swap）的含义和作用
- 了解脏页的含义和作用

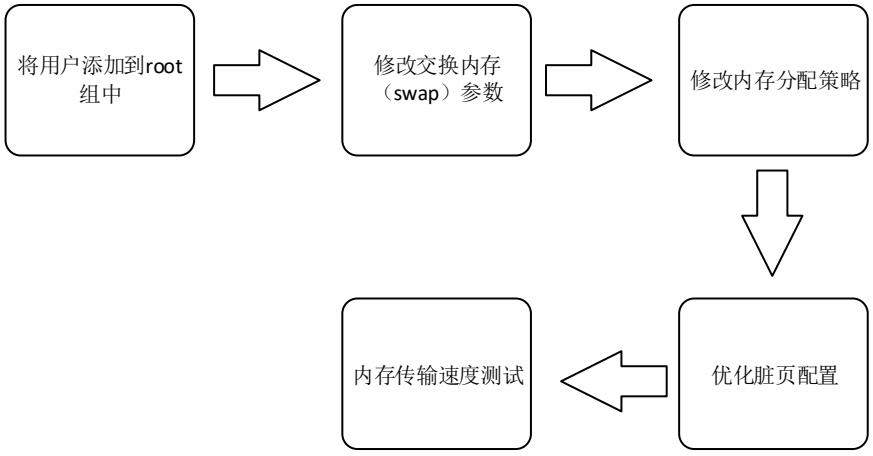
1.3. 实验环境

本实验所需之主要资源环境如表 1-1 所示。

表 1-1 资源环境

服务器集群	3 个以上节点，节点间网络互通，各节点最低配置：双核 CPU、8GB 内存、100G 硬盘
运行环境	CentOS 7.4 （gui 英文版本）
用户名/密码	root/password hadoop/password
服务和组件	HDFS、YARN、MapReduce 等，其他服务根据实验需求安装

1.4. 实验视图



1.5. 实验过程

1.5.1. 实验任务一：将 Hadoop 用户添加到 root 组中

1.5.1.1. 步骤一：切换到 root 账号

本实验需要修改的配置文件属于 root 用户所有，所以我们需要先切换到 root 账号。root 账号的密码是 password

[hadoop@master /]$ su root

1.5.1.2. 步骤二：查看/etc/sudoers 文件的读写权限

使用 ls -l 命令查看/etc/sudoers 文件的读写权限

[root@master /]# ls -l /etc/sudoers

-r-r-----. 1 root root 3938 Jul 6 18:44 /etc/sudoers

可以看出，/etc/sudoers 文件对 root 账号和 root 用户组只有读的权限对于其他用户没有任何权限

1.5.1.3. 步骤三：给/etc/sudoers 文件添加可写权限

给/etc/sudoers 文件的所有者 root 账户添加可写权限

[root@master hadoop]# chmod u+w /etc/sudoers

再次使用 ls -l 命令查看/etc/sudoers 文件的读写权限

[root@master hadoop]# ls -l /etc/sudoers

-rw--r---. 1 root root 3938 Jul 6 18:44 /etc/sudoers

1.5.1.4. 步骤四：用 vi 打开/etc/sudoers 文件

[root@master ~]# vi /etc/sudoers

1.5.1.5. 步骤五：添加配置信息

找到 root ALL=(ALL) ALL 这一行，在后面再加上一行 hadoop ALL=(ALL) ALL

按 Esc 键，在按:wq，保存并退出

1.5.1.6. 步骤六：恢复/etc/sudoers 文件的读写权限

将 sudoers 文件的可写权限去掉，恢复到初始状态

[root@master ~]# chmod u-w /etc/sudoers

进入 Hadoop 用户

[root@master ~]# su hadoop

[root@master root]# cd

1.5.1.7. 步骤七：三台虚拟机同步操作

在 slave1 和 slave2 上也进行以上操作，同时本章中的所有提到的参数均需要在三台虚拟机上进行修改。

1.5.2. 实验任务二：避免使用 swap 分区

1.5.2.1. 步骤一：查看 swappiness 的值

使用 cat 命令查看 swappiness 的值

[hadoop@master ~]$ cat /proc/sys/vm/swappiness

可以看到初始值为 30

1.5.2.2. 步骤二：修改配置文件

使用 vi 打开配置文件，修改配置文件
[hadoop@master ~]$ sudo vi /etc/sysctl.conf
找到 vm.swappiness 参数并更改其值为 1。如果此参数不存在，请将以下行附加到该文件 /etc/sysctl.conf 中
vm.swappiness = 1
按 Esc 键，再按:wq，保存并退出

1.5.2.3. 步骤三：使配置生效

[hadoop@master ~]$ sudo sysctl -p

再次使用[hadoop@master ~]$ cat /proc/sys/vm/swappiness 看参数是否生效
如果参数生效应该可以看到返回结果为 1

1.5.3. 实验任务三：调整内存分配策略

1.5.3.1. 步骤一： 查看当前 vm.overcommit_memory 参数的值

使用命令 [hadoop@master ~]$ sysctl -n vm.overcommit_memory 来查看当前 vm.overcommit_memory 的参数值
可以看到当前的 vm.overcommit_memory 的参数值为 0

1.5.3.2. 步骤二： 永久性修改 vm.overcommit_memory 参数的值

使用 vi 打开/etc/sysctl.conf 文件,在文件末尾加入 vm.overcommit_memory=2 参数并保存
[hadoop@master ~]$ sudo vim /etc/sysctl.conf
vm.overcommit_memory=2
按 Esc 键，再按:wq，保存并退出
Tips:将 vm.overcommit_memory 的参数如果修改为 2 的话会导致无法进入图形界面，如果需要使用系统的图形界面可以将 vm.overcommit_memory 的参数值修改为 1

1.5.3.3. 步骤三：使参数生效

[hadoop@master ~]$ sudo sysctl -p

再次使用命令[hadoop@master ~]$ sysctl -n vm.overcommit_memory 验证参数是否生效
如果参数生效应该可以看到返回值为 2

1.5.4. 实验任务四：脏页配置优化

1.5.4.1. 步骤一：查看当前脏页相关配置信息

查看当前脏页配置信息
[hadoop@master ~]$ sysctl -a | grep dirty
可以看到返回的脏页参数为：
vm.dirty_background_bytes = 0
vm.dirty_background_ratio = 10
vm.dirty_bytes = 0
vm.dirty_expire_centisecs = 3000

vm.dirty_ratio = 30
vm.dirty_writeback_centisecs = 500

1.5.4.2. 步骤二：修改脏页配置信息

通过修改 vm.dirty_background_ratio 的值和 vm.dirty_radio 的值来优化脏页配置

[hadoop@master ~]$ sudo vi /etc/sysctl.conf

在文件末尾添加两个参数

vm.dirty_background_ratio=5

vm.dirty_ratio=80

按 Esc 键，再按:wq，保存并退出

1.5.4.3. 步骤三：使参数生效

[hadoop@master ~]$ sudo sysctl -p

执行命令后的返回值应该为修改的参数的数据

vm.dirty_background_ratio = 5

vm.dirty_ratio = 80

1.5.5. 实验任务五：内存读写速度测试

1.5.5.1. 步骤一：使用 linux 自带命令测试内存读写速度

可以使用以下命令对内存进行读写速度进行测试
[hadoop@master /]$ dd if=/dev/zero of=/dev/null bs=1M count=1024
其中 if 代表输入文件位置一般都填写默认值，of 代表输出文件位置，因为是对内存进行读写所以也可以采用默认值，bs 的含义是同时设置读入/输出的块大小为 bytes 个字节。count 代表进行几次读写
返回数据的格式为：
记录了 1024+0 的读入
记录了 1024+0 的写出
1073741824 字节(1.1 GB)已复制，0.066381 秒，16.2 GB/秒 18.3GB/s 代表的就是当前的内存读写速度

2. 实验二：优化 Linux 系统网络

2.1. 实验目的

完成本实验，您应该能够：

- 掌握 Linux 平台防火墙操作
- 掌握通过命令禁用 ipv6
- 掌握修改 somaxconn 以限制 tcp 连接侦听队列大小
- 掌握使用通过 sysctl 命令调整分配给读写缓冲区的内存大小

2.2. 实验要求

- 熟悉常用 Linux 操作系统命令
- 了解缓冲区对 Linux 读写的意义
- 了解 Linux 网络

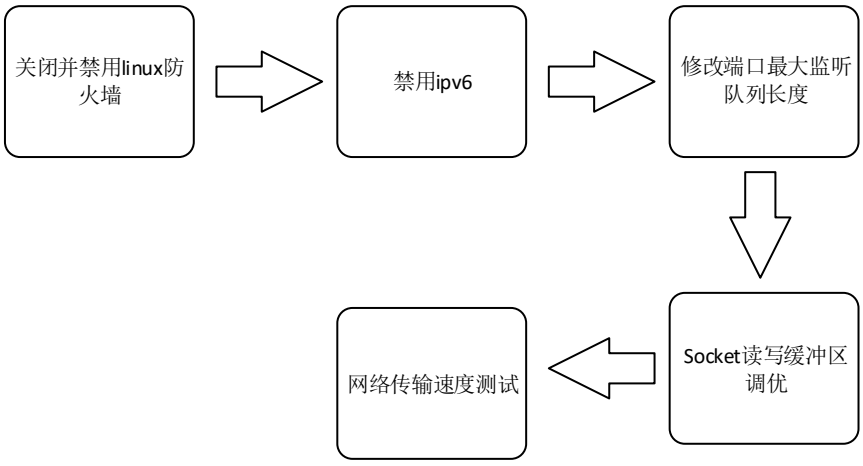
2.3. 实验环境

本实验所需之主要资源环境如表 1-1 所示。

表 1-1 资源环境

服务器集群	3 个以上节点，节点间网络互通，各节点最低配置：双核 CPU、8GB 内存、100G 硬盘
运行环境	CentOS 7.4 （gui 英文版本）
用户名/密码	root/password hadoop/password
服务和组件	Iperf3.1.1

2.4. 实验视图



2.5. 实验过程

2.5.1. 实验任务一：关闭 Linux 防火墙

2.5.1.1. 步骤一：使用命令查看 Linux 防火墙状态

打开终端输入 sudo systemctl status firewalld 查看防火墙状态，如图 1-1 所示

[hadoop@master ~]$ sudo systemctl status firewalld

状态如下所示

● firewalld.service - firewalld - dynamic firewall daemon
Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
Active: active (running) since Tue 2020-07-07 00:36:11 PDT; 5s ago
Docs: man:firewalld(1)
Main PID: 12618 (firewalld)
CGroup: /system.slice/firewalld.service
└─12618 /usr/bin/python -Es /usr/sbin/firewalld --nofork --nopid

2.5.1.2. 步骤二：使用命令关闭防火墙

在终端中输入 sudo systemctl stop firewalld 以关闭防火墙，如图 1-2 所示

[hadoop@master ~]$ sudo systemctl stop firewalld

在终端中输入 sudo systemctl status firewalld 再次查看防火墙状态，如图 1-3 所示

[hadoop@master ~]$ sudo systemctl status firewalld

状态如下所示

● firewalld.service - firewalld - dynamic firewall daemon
Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
Active: inactive (dead)
Docs: man:firewalld(1)

2.5.1.3. 步骤三：禁止开机启动防火墙

在终端中输入 sudo systemctl disable firewalld 命令禁止开机启动防火墙，如图 1-4 所示

[hadoop@master ~]$ sudo systemctl disable firewalld

2.5.2. 实验任务二： 禁用 ipv6

2.5.2.1. 步骤一：修改配置文件

在终端中输入 sudo vi /etc/sysctl.conf

[hadoop@master ~]$ sudo vi /etc/sysctl.conf

在文件末尾添加 net.ipv6.conf.all.disable_ipv6=1 保存

2.5.2.2. 步骤二：使配置生效

在终端中输入 sudo sysctl -p 使配置生效

[hadoop@master ~]$ sudo sysctl -p

在返回值中显示 net.ipv6.conf.all.disable_ipv6 = 1 说明参数配置生效

2.5.3. 实验任务三：修改端口最大监听队列长度

2.5.3.1. 步骤一：查看当前监听队列大小

打开终端，输入 sudo cat /etc/sysctl.conf (若无 net.core.somaxconn 项则手动添加)
[hadoop@master ~]$ sudo cat /etc/sysctl.conf

2.5.3.2. 步骤二：修改监听队列长度

在终端中输入 sudo vi /etc/sysctl.conf
[hadoop@master ~]$ sudo vi /etc/sysctl.conf
在文件末尾追加 net.core.somaxconn=32768
如果已经存在 net.core.somaxconn 参数则修改它的的值为 32768

2.5.3.3. 步骤三：使配置生效

在终端中输入 sudo sysctl -p 使配置生效
[hadoop@master ~]$ sudo sysctl -p
当返回值中存在 net.core.somaxconn = 32768 说明配置生效

2.5.4. 实验任务四： socket 读写缓冲区调优

2.5.4.1. 步骤一：设置 tcp 数据发送窗口大小为 256kb

在终端中输入 sudo sysctl -q net.core.wmem_default
[hadoop@master ~]$ sudo sysctl -q net.core.wmem_default
在终端中输入 echo "net.core.wmem_default=256960"|sudo tee -a /etc/sysctl.conf
[hadoop@master ~]$ echo "net.core.wmem_default=256960"|sudo tee -a /etc/sysctl.conf
最后在终端中输入 sudo sysctl -p 使配置生效
[hadoop@master ~]$ sudo sysctl -p
再次使用命令[hadoop@master ~]$ sudo sysctl -q net.core.wmem_default
若返回值为 net.core.wmem_default=256960 说明配置生效

2.5.4.2. 步骤二：设置 tcp 数据接收窗口大小为 256kb

在终端中输入 sysctl -q net.core.rmem_default
[hadoop@master ~]$ sysctl -q net.core.rmem_default
在终端中输入 echo " net.core.rmem_default=256960"|sudo tee -a /etc/sysctl.conf
[hadoop@master ~]$ echo " net.core.rmem_default=256960"|sudo tee -a /etc/sysctl.conf
最后在终端中输入 sudo sysctl -p 使配置生效
[hadoop@master ~]$ sudo sysctl -p
再次使用命令[hadoop@master ~]$ sysctl -q net.core.rmem_default
若返回值为 net.core.rmem_default=256960 说明配置生效

2.5.4.3. 步骤三：设置最大 TCP 数据发送缓冲区最大值为 2M

在终端中输入 sysctl -q net.core.wmem_max
[hadoop@master ~]$ sysctl -q net.core.wmem_max
在终端中输入 echo "net.core.wmem_max=2097152"| sudo tee -a /etc/sysctl.conf

[hadoop@master ~]$ echo "net.core.wmem_max=2097152"| sudo tee -a /etc/sysctl.conf
最后在终端中输入 sudo sysctl -p 使配置生效
[hadoop@master ~]$ sudo sysctl -p
再次使用命令[hadoop@master ~]$ sysctl -q net.core.wmem_max
若返回值为 net.core.wmem_max=2097152 说明配置生效

2.5.4.4. 步骤四：设置最大 TCP 数据接收缓冲区最大值为 2M

在终端中输入 sysctl -q net.core.rmem_max
[hadoop@master ~]$ sysctl -q net.core.rmem_max
在终端中输入 echo " net.core.rmem_max=2097152"| sudo tee -a /etc/sysctl.conf
[hadoop@master ~]$ echo " net.core.rmem_max=2097152"| sudo tee -a /etc/sysctl.conf
最后在终端中输入 sudo sysctl -p 使配置生效
[hadoop@master ~]$ sudo sysctl -p
再次使用命令[hadoop@master ~]$ sysctl -q net.core.rmem_max
若返回值为 net.core.rmem_max=2097152 说明配置生效

2.5.5. 实验任务五：网络传输速度测试

2.5.5.1. 步骤一：安装测试工具 iperf

首先解压 iperf
[hadoop@master ~]$ cd /usr/local/src
[hadoop@master src]$ sudo tar -zxvf /opt/software/iperf-3.1.1-source.tar.gz
然后进入 iperf 文件
[hadoop@master src]$ cd iperf-3.1.1/
然后依次执行以下三条命令
[hadoop@master iperf-3.1.1]$ sudo ./configure
[hadoop@master iperf-3.1.1]$ sudo make
[hadoop@master iperf-3.1.1]$ sudo make install
同时在 slave1 以及 slave2 上以相同的步骤安装 iperf
[hadoop@master iperf-3.1.1]$ sudo scp /opt/software/iperf-3.1.1-source.tar.gz slave1:/root
[hadoop@master iperf-3.1.1]$ sudo scp /opt/software/iperf-3.1.1-source.tar.gz slave2:/root
[hadoop@slave1~]$ cd /usr/local/src
[hadoop@slave1 src]$ sudo tar -zxvf /root/iperf-3.1.1-source.tar.gz
[hadoop@slave1 src]$ cd iperf-3.1.1/
然后依次执行以下三条命令
[hadoop@slave1 iperf-3.1.1]$ sudo ./configure
[hadoop@slave1 iperf-3.1.1]$ sudo make
[hadoop@slave1 iperf-3.1.1]$ sudo make install

[hadoop@slave2~]$ cd /usr/local/src
[hadoop@slave2 src]$ sudo tar -zxvf /root/iperf-3.1.1-source.tar.gz
[hadoop@slave2 src]$ cd iperf-3.1.1/

然后依次执行以下三条命令

```
[hadoop@slave2 iperf-3.1.1]$ sudo ./configure
[hadoop@slave2 iperf-3.1.1]$ sudo make
[hadoop@slave2 iperf-3.1.1]$ sudo make install
```

2.5.5.2. 步骤二：测试 master 与 slave1 之间的网络传输速度

在 master 上开启测试端口 5201

```
[hadoop@master iperf-3.1.1]$ iperf3 -s
-----------------------------------------------------------
Server listening on 5201
```

使用命令[hadoop@slave1 iperf-3.1.1]$ iperf3 -c 192.168.1.6 测试 slave1 到 master 的传输速度

```
[hadoop@slave1 iperf-3.1.1]$ iperf3 -c 192.168.1.6
Connecting to host 192.168.1.6, port 5201
[  4] local 192.168.1.7 port 53084 connected to 192.168.1.6 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   499 MBytes  4.19 Gbits/sec    0   1.13 MBytes
[  4]   1.00-2.00   sec   526 MBytes  4.41 Gbits/sec    0   1.20 MBytes
[  4]   2.00-3.00   sec   604 MBytes  5.06 Gbits/sec    0   1.33 MBytes
[  4]   3.00-4.00   sec   596 MBytes  5.00 Gbits/sec    0   1.41 MBytes
[  4]   4.00-5.00   sec   591 MBytes  4.96 Gbits/sec    0   1.46 MBytes
[  4]   5.00-6.00   sec   608 MBytes  5.10 Gbits/sec    0   1.50 MBytes
[  4]   6.00-7.00   sec   601 MBytes  5.04 Gbits/sec    0   1.53 MBytes
[  4]   7.00-8.00   sec   588 MBytes  4.93 Gbits/sec    0   1.56 MBytes
[  4]   8.00-9.00   sec   600 MBytes  5.04 Gbits/sec    0   1.57 MBytes
[  4]   9.00-10.00  sec   590 MBytes  4.95 Gbits/sec    0   1.59 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  5.67 GBytes  4.87 Gbits/sec    0             sender
[  4]   0.00-10.00  sec  5.66 GBytes  4.86 Gbits/sec                  receiver

iperf Done.
```

测试结果中的 Bandwidth 就是两台服务器之间的传输速度

3. 实验三 优化 Linux 系统磁盘

3.1. 实验目的

完成本实验，您应该能够：

- 了解三种 I/O 调度器的优缺点并自主选择最优的调度器
- 了解时间戳对 Linux I/O 的影响

3.2. 实验要求

- 了解 I/O 调度器对 linux 的作用
- 了解文件记录访问时间戳的意义

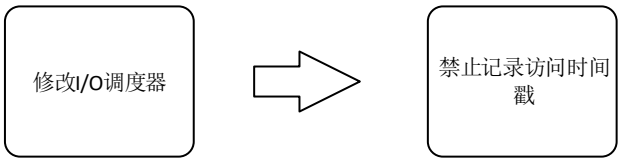
3.3. 实验环境

本实验所需之主要资源环境如表 1-1 所示。

表 1−1 资源环境

服务器集群	3 个以上节点，节点间网络互通，各节点最低配置：双核 CPU、8GB 内存、100G 硬盘
运行环境	CentOS 7.4 （gui 英文版本）
用户名/密码	root/password hadoop/password
服务和组件	Linux

3.4. 实验视图



3.5. 实验过程

3.5.1. 实验任务一：修改 I/O 调度器

3.5.1.1. 步骤一：查看当前系统支持的 I/O 调度器

在终端中输入命令 dmesg | grep -i scheduler，如图 2-1 所示

[hadoop@master iperf-3.1.1]$ cd

[hadoop@master ~]$ sudo dmesg | grep -i scheduler

返回值为

[8.605539] io scheduler noop registered

[8.605542] io scheduler deadline registered (default)

[8.605572] io scheduler cfq registered

3.5.1.2. 步骤二：查看硬盘的 IO 调度算法 I/O 调度器

终端中时输入命令 cat /sys/block/sda/queue/scheduler，如图 2-1 所示

[hadoop@master ~]$ cat /sys/block/sda/queue/scheduler

返回值为

noop deadline [cfq]

3.5.1.3. 步骤三：临时修改当前 I/O 调度器

在终端中输入 sudo chmod o+wr /sys/block/sda/queue/scheduler 给用户添加权限

[hadoop@master ~]$ sudo chmod o+w /sys/block/sda/queue/scheduler

在终端中输入 echo noop > /sys/block/sda/queue/scheduler

[hadoop@master ~]$ sudo echo noop > /sys/block/sda/queue/scheduler

在终端中输入 cat /sys/block/sda/queue/scheduler 查看该文件

[hadoop@master ~]$ cat /sys/block/sda/queue/scheduler

[noop] deadline cfq

在终端中输入 sudo chmod o-wr /sys/block/sda/queue/scheduler 回收用户权限

[hadoop@master ~]$ sudo chmod o-wr /sys/block/sda/queue/scheduler

3.5.1.4. 步骤四：永久修改当前 I/O 调度器

在终端中输入 sudo grubby --update-kernel=ALL --args="elevator=noop"

[hadoop@master ~]$ sudo grubby --update-kernel=ALL --args="elevator=noop"

重启后终端输入 cat /sys/block/sda/queue/scheduler

[hadoop@master ~]$ sudo reboot

[root@master ~]# su hadoop

[hadoop@master root]$ cd

[hadoop@master ~]$ cat /sys/block/sda/queue/scheduler

[noop] deadline cfq

Tips:将 I/O 调度器调整为 noop 是最适合虚拟机在固态硬盘中的用户，对于机械硬盘的用户，一般来说 centos7 的默认 I/O 调度器已经为 deadline 所以无需调整

3.5.2. 实验任务二：禁止记录访问时间戳

3.5.2.1. 步骤一：修改/etc/fstab 文件

在终端中输入命令 sudo vi /etc/fstab

[hadoop@master ~]$ sudo vi /etc/fstab

在 defaults 后面添加“noatime,nodiratime”表示不记录文件访问时间

```
#
# /etc/fstab
# Created by anaconda on Wed May 20 01:10:47 2020
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
UUID=9744bd7c-41db-47e8-b218-60f3a3d842f4   /                       xfs
defaults,noatime,nodiratime             0 0
UUID=c1d20982-23c1-4974-9643-27cccd72eaae   /boot                   xfs
defaults            0 0
UUID=a1f33604-7437-4da0-b4ee-e08ef73bc514   swap                    swap
defaults            0 0
```

3.5.2.2. 步骤二：使配置生效

方法一：重启系统

方法二：在终端中输入命令：mount -o remount /

[hadoop@master ~]$ sudo mount -o remount /

4. 实验四 优化 Linux 文件系统

4.1. 实验目的

完成本实验，您应该能够：

- 掌握增大可打开文件描述符数目方法
- 掌握关闭透明大页面的方法
- 掌握关闭 selinux 的方法

4.2. 实验要求

- 熟悉常用 Linux 操作系统命令
- 了解 THP 在 linux 系统中的作用
- 了解 selinux 在 linux 系统中的作用

4.3. 实验环境

本实验所需之主要资源环境如表 1-1 所示。

表 1−1 资源环境

服务器集群	3 个以上节点，节点间网络互通，各节点最低配置：双核 CPU、8GB 内存、100G 硬盘
运行环境	CentOS 7.4 （gui 英文版本）
用户名/密码	root/password hadoop/password
服务和组件	Linux

4. 4. 实验视图

4.5. 试验过程

4.5.1. 实验任务一：增大可打开文件描述符的数目

4.5.1.1. 步骤一：修改 limits.conf

在终端中输入 sudo vi /etc/security/limits.conf

[hadoop@master ~]$ sudo vi /etc/security/limits.conf

在文件中的#@student 和#End of fIle 中间添加以下内容

#@student - maxlogins 4

* hard nofile 1048576

* soft nproc 1048576

* hard nproc 1048576

* soft memlock unlimited

* hard memlock unlimited

#End of file

保存并退出

4.5.1.2. 步骤二：修改 20-nproc.conf

在终端中输入 sudo vi /etc/security/limits.d/20-nproc.conf

[hadoop@master ~]$ sudo vi /etc/security/limits.d/20-nproc.conf

将第一列为*的用户的限制 4096 修改为 1048576

* soft nproc 1048576

root soft nproc unlimited

重启后查看当前最大打开文件数

[hadoop@master ~]$ sudo reboot

[root@master ~]# su hadoop

[hadoop@master root]$ cd

[hadoop@master ~]$ ulimit -u

返回值为 1048576 说明参数修改成功

4.5.2. 实验任务二：关闭 THP

4.5.2.1. 步骤一：查询 linux 内核版本以及透明大页面的状态

在终端中输入 cat /sys/kernel/mm/transparent_hugepage/enabled

[hadoop@master ~]$ cat /sys/kernel/mm/transparent_hugepage/enabled

返回值为

[always] madvise never

4.5.2.2. 步骤二：关闭透明大页面

在终端中输入 sudo vi /etc/default/grub

[hadoop@master ~]$ sudo vi /etc/default/grub

修改 GRUB_CMDLINE_LINUX="crashkernel=auto rhgb quiet"为 GRUB_CMDLINE_LINUX="crashkernel=auto rhgb quiet transparent_hugepage=never"

4.5.2.3. 步骤三：应用修改

在终端中输入 sudo grub2-mkconfig -o /boot/grub2/grub.cfg

[hadoop@master ~]$ sudo grub2-mkconfig -o /boot/grub2/grub.cfg

重启后在终端输入 cat /sys/kernel/mm/transparent_hugepage/enabled

[hadoop@master ~]$ sudo reboot

[root@master ~]# su hadoop

[hadoop@master root]$ cd

[hadoop@master ~]$ cat /sys/kernel/mm/transparent_hugepage/enabled

always madvise [never]

4.5.3. 实验任务三：关闭 SELinux

4.5.3.1. 步骤一：查看 SELinux 的状态

[hadoop@master ~]$ getenforce

Enforcing

若显示为 Enforcing 说明 SELinux 服务已经启动，若为 Disabled 说明 SELinux 已经关闭。

4.5.3.2. 步骤二：关闭 SELinux

在终端中输入 sudo vi /etc/selinux/config

[hadoop@master ~]$ sudo vi /etc/selinux/config

将 SELINUX=enforcing 改为 SELINUX=disabled

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=disabled

SELINUXTYPE= can take one of three two values:

targeted - Targeted processes are protected,

minimum - Modification of targeted policy. Only selected processes are protected.

mls - Multi Level Security protection.

SELINUXTYPE=targeted

重启

在终端中输入 /usr/sbin/sestatus -v 查看 SELinux 的状态

```
[hadoop@master ~]$ /usr/sbin/sestatus -v
SELinus status:    disable
```

5. 实验五 优化 Linux 系统缓冲区

5.1. 实验目的

完成本实验，您应该能够：

- 掌握优化 Linux 文件系统缓冲区的方法
- 掌握使用 hadoop 自带程序进行性能测试

5.2. 实验要求

- 了解预读缓冲区对 Linux 系统的作用
- 了解常用的 hadoop 集群操作

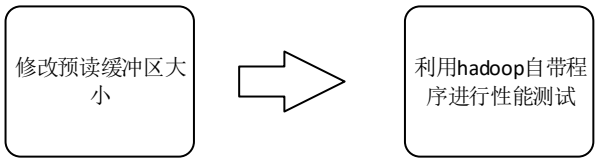
5.3. 实验环境

本实验所需之主要资源环境如表 1-1 所示。

表 1-1 资源环境

服务器集群	3 个以上节点，节点间网络互通，各节点最低配置：双核 CPU、8GB 内存、100G 硬盘
运行环境	CentOS 7.4 （gui 英文版本）
用户名/密码	root/password hadoop/password
服务和组件	Linux, Hadoop2.7

5.4. 实验视图



5.5. 实验过程

5.5.1. 实验任务一：设置合理的预读取缓冲区大小

5.5.1.1. 步骤一：查看磁盘占用情况

在终端中输入 df -h

```
[hadoop@master ~]$ df -h
3.8G        0  3.8G   0% /dev
tmpfs           3.9G     0  3.9G   0% /dev/shm
tmpfs           3.9G  9.0M  3.9G   1% /run
tmpfs           3.9G     0  3.9G   0% /sys/fs/cgroup
/dev/sda1       297M  157M  141M  53% /boot
```

```
tmpfs           781M  4.0K  781M   1% /run/user/42
tmpfs           781M   20K  781M   1% /run/user/1000
```

5.5.1.2. 步骤二：读取设备的预读值

在终端中输入 blockdev --getra /dev/sda1

```
[hadoop@master ~]$ sudo blockdev --getra /dev/sda1
```

返回值为 8192

5.5.1.3. 步骤三：修改设备的预读值

在终端中输入 blockdev --setra 10240 /dev/sda1 以修改预读值为 10240

```
[hadoop@master ~]$ sudo blockdev --setra 10240 /dev/sda1
```

5.5.1.4. 步骤四：再次查看预读值

在终端中输入 blockdev --getra /dev/sda1

```
[hadoop@master ~]$ sudo blockdev --getra /dev/sda1
```

返回值为 10240 说明参数修改成功

5.5.2. 实验任务二：利用 hadoop 自带测试程序测试磁盘性能

修改了磁盘，文件系统以及文件缓冲区之后可以利用 hadoop 自带的测试程序来测试磁盘的传输性能

5.5.2.1. 步骤一：启动 hadoop 集群

在 master，slave1 以及 slave2 上启动 zookeeper

```
[hadoop@master ~]$ zkServer.sh start
[hadoop@slave1 ~]$ zkServer.sh start
[hadoop@slave2 ~]$ zkServer.sh start
```

在 master 上直接启动集群

```
[hadoop@master ~]$ start-all.sh
```

5.5.2.2. 步骤二:进行测试

进入到/usr/local/src/hadoop/share/hadoop/mapreduce/下

```
[hadoop@master ~]$ cd /usr/local/src/hadoop/share/hadoop/mapreduce/
```

利用命令进行写入文件测试

```
[hadoop@master mapreduce]$ hadoop jar hadoop-mapreduce-client-jobclient-2.7.1-tests.jar TestDFSIO -write -size 1GB
```

测试结果

```
20/08/05 09:04:29 WARN hdfs.DFSClient: DFSInputStream has been closed already
20/08/05 09:04:29 INFO fs.TestDFSIO: ----- TestDFSIO ----- : write
20/08/05 09:04:29 INFO fs.TestDFSIO:                Date & time: Wed Aug 05 09:04:29 CST 2020
20/08/05 09:04:29 INFO fs.TestDFSIO:            Number of files: 1
20/08/05 09:04:29 INFO fs.TestDFSIO: Total MBytes processed: 1024.0
20/08/05 09:04:29 INFO fs.TestDFSIO:         Throughput mb/sec: 9.683947721813471
20/08/05 09:04:29 INFO fs.TestDFSIO: Average IO rate mb/sec: 9.683947563171387
20/08/05 09:04:29 INFO fs.TestDFSIO:    IO rate std deviation: 0.0011379476762855877
20/08/05 09:04:29 INFO fs.TestDFSIO:        Test exec time sec: 131.786
20/08/05 09:04:29 INFO fs.TestDFSIO: 利用命令进行读取文件测试
```

[hadoop@master mapreduce]$ hadoop jar hadoop-mapreduce-client-jobclient-2.7.1-tests.jar TestDFSIO -read -size 1GB
20/08/05 09:06:52 WARN hdfs.DFSClient: DFSInputStream has been closed already
20/08/05 09:06:52 INFO fs.TestDFSIO: ----- TestDFSIO ----- : read
20/08/05 09:06:52 INFO fs.TestDFSIO: Date & time: Wed Aug 05 09:06:52 CST 2020
20/08/05 09:06:52 INFO fs.TestDFSIO: Number of files: 1
20/08/05 09:06:52 INFO fs.TestDFSIO: Total MBytes processed: 1024.0
20/08/05 09:06:52 INFO fs.TestDFSIO: Throughput mb/sec: 15.166550646503843
20/08/05 09:06:52 INFO fs.TestDFSIO: Average IO rate mb/sec: 15.166550636291504
20/08/05 09:06:52 INFO fs.TestDFSIO: IO rate std deviation: 0.0018087053803741643
20/08/05 09:06:52 INFO fs.TestDFSIO: Test exec time sec: 98.047
20/08/05 09:06:52 INFO fs.TestDFSIO:

利用命令删除生成的测试数据

[hadoop@master mapreduce]$ hadoop jar hadoop-mapreduce-client-jobclient-2.7.1-tests.jar TestDFSIO -clean

如果想进行多文件的测试可以将后面的参数进行修改

[hadoop@master mapreduce]$ hadoop jar hadoop-mapreduce-client-jobclient-2.7.1-tests.jar TestDFSIO -write -nrFiles 10 -fileSize 128MB

例如这条命令是用与测试 10 个 128MB 文件写入速度，-nrFiles 为指定文件数，-fileSize 为指定每个文件的大小，读取同理