-

目录

[1. 实验一：查看大数据平台日志信息 2](#_Toc26427)

[1.1. 实验目的 2](#_Toc24487)

[1.2. 实验要求 2](#_Toc28827)

[1.3. 实验环境 2](#_Toc14083)

[1.4. 实验过程 2](#_Toc46)

[1.4.1. 实验任务一：查看大数据平台主机日志 2](#_Toc30453)

[1.4.2. 实验任务二： 在Hadoop MapReduce Jobs中查看日志信息 9](#_Toc25981)

[1.4.3. 实验任务三：通过用户界面查看Hadoop日志 11](#_Toc1089)

[1.4.4. 实验任务四：通过命令查看Hadoop日志 14](#_Toc16)

[1.4.5. 实验任务五：查看HBase日志 15](#_Toc32447)

[1.4.6. 实验任务六：查看HBase日志 16](#_Toc32013)

[2. 实验二 查看大数据平台告警信息 18](#_Toc24835)

[2.1. 实验目的 18](#_Toc28336)

[2.2. 实验要求 18](#_Toc22120)

[2.3. 实验环境 18](#_Toc29879)

[2.4. 实验过程 18](#_Toc16986)

[2.4.1. 实验任务一：查看大数据平台主机告警信息 18](#_Toc9458)

[2.4.2. 实验任务二：查看Hadoop告警信息 19](#_Toc2836)

[2.4.3. 实验任务三：查看HBase告警信息 22](#_Toc7580)

[2.4.4. 实验任务四：查看Hive告警信息 25](#_Toc2863)

# 实验一：查看大数据平台日志信息

## 实验目的

完成本实验，您应该能够：

* 掌握大数据平台日志和告警的查看
* 掌握常见告警问题的解决方法

## 实验要求

* 了解大数据平台日志信息
* 了解大数据平台告警信息

## 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 |
| 大数据平台 | H3C DataEngine-E0104-RHEL6-X86\_64 |
| 服务和组件 | 完成前面章节的实验，其他服务及组件根据实验需求安装 |

## 实验过程

### 实验任务一：查看大数据平台主机日志

Linux 操作系统本身和大部分服务器程序的日志文件都默认放在目录/var/log/下。一部分程序共用一个日志文件，一部分程序使用单个日志文件，而有些大型服务器程序由于日志文件不止一个，所以会在/var/log/目录中建立相应的子目录来存放日志文件，这样既保证了日志文件目录的结构清晰，又可以快速定位日志文件。有相当一部分日志文件只有 root用户才有权限读取，这保证了相关日志信息的安全性。

使用hadoop用户登录Linux主机，切换到/var/log目录，执行ll命令查询该目录所有日志文件。

[hadoop@master ~]$ cd /var/log

[hadoop@master log]$ ll

**总用量 2272**

**drwxr-xr-x. 2 root root 204 4月 23 18:57 anaconda**

**drwx------. 2 root root 23 4月 23 18:57 audit**

**-rw-------. 1 root root 0 5月 24 03:17 boot.log**

**-rw------- 1 root root 0 5月 18 03:11 boot.log-20200518**

**-rw------- 1 root root 0 5月 19 03:40 boot.log-20200519**

**-rw------- 1 root root 0 5月 20 03:26 boot.log-20200520**

**-rw------- 1 root root 0 5月 21 03:47 boot.log-20200521**

**-rw------- 1 root root 0 5月 22 03:27 boot.log-20200522**

**-rw------- 1 root root 0 5月 23 03:21 boot.log-20200523**

**-rw------- 1 root root 0 5月 24 03:17 boot.log-20200524**

**-rw------- 1 root utmp 1920 5月 24 12:40 btmp**

**-rw-------. 1 root utmp 0 4月 23 18:54 btmp-20200504**

**drwxr-xr-x. 2 chrony chrony 6 8月 4 2017 chrony**

**-rw------- 1 root root 3152 5月 24 15:01 cron**

**-rw------- 1 root root 11874 5月 4 21:29 cron-20200504**

**-rw------- 1 root root 31449 5月 10 03:43 cron-20200510**

**-rw------- 1 root root 54034 5月 18 03:11 cron-20200518**

**-rw------- 1 root root 40445 5月 24 03:17 cron-20200524**

**-rw-r--r-- 1 root root 124149 5月 9 09:12 dmesg**

**-rw-r--r-- 1 root root 123941 5月 7 21:57 dmesg.old**

**-rw-r--r--. 1 root root 1842 5月 24 13:21 firewalld**

**-rw-r--r--. 1 root root 193 4月 23 18:54 grubby\_prune\_debug**

**-rw-r--r--. 1 root root 292584 5月 24 15:09 lastlog**

**-rw------- 1 root root 0 5月 24 03:17 maillog**

**-rw------- 1 root root 585296 4月 27 21:43 maillog-20200504**

**-rw------- 1 root root 0 5月 4 21:29 maillog-20200510**

**-rw------- 1 root root 0 5月 10 03:43 maillog-20200518**

**-rw------- 1 root root 0 5月 18 03:11 maillog-20200524**

**-rw------- 1 root root 3898 5月 24 15:09 messages**

**-rw------- 1 root root 141627 5月 4 21:19 messages-20200504**

**-rw------- 1 root root 435302 5月 10 03:01 messages-20200510**

**-rw------- 1 root root 29663 5月 18 03:01 messages-20200518**

**-rw------- 1 root root 22282 5月 24 03:01 messages-20200524**

**-rw-r--r-- 1 mysql mysql 85512 5月 24 15:10 mysqld.log**

**drwxr-xr-x. 2 root root 6 4月 23 18:57 rhsm**

**-rw------- 1 root root 528 5月 24 12:41 secure**

**-rw------- 1 root root 1325 5月 4 21:19 secure-20200504**

**-rw------- 1 root root 22121 5月 9 09:32 secure-20200510**

**-rw------- 1 root root 7065 5月 16 00:52 secure-20200518**

**-rw------- 1 root root 3169 5月 23 18:03 secure-20200524**

**-rw------- 1 root root 0 5月 24 03:17 spooler**

**-rw------- 1 root root 0 4月 26 03:50 spooler-20200504**

**-rw------- 1 root root 0 5月 4 21:29 spooler-20200510**

**-rw------- 1 root root 0 5月 10 03:43 spooler-20200518**

**-rw------- 1 root root 0 5月 18 03:11 spooler-20200524**

**-rw-------. 1 root root 0 4月 23 18:53 tallylog**

**drwxr-xr-x. 2 root root 23 4月 23 18:57 tuned**

**-rw-r--r--. 1 root root 9978 5月 9 09:12 vmware-vgauthsvc.log.0**

**-rw-r--r--. 1 root root 20397 5月 9 09:12 vmware-vmsvc.log**

**-rw-rw-r--. 1 root utmp 31488 5月 24 13:19 wtmp**

**-rw------- 1 root root 1728 4月 23 12:56 yum.log**

结果显示，包含了以下多种功能的日志文件，下面逐一查看这些日志内容。

#### 步骤一：查看内核及公共消息日志（/var/log/messages）。

内核及公共信息日志是许多进程日志文件的汇总，可以切换到root用户，采用cat或tail命令查看该文件。

[hadoop@master log]$ su root

**password：**

[root@master log]# cat messages

**May 24 03:17:01 master rsyslogd: [origin software="rsyslogd" swVersion="8.24.0" x-pid="679" x-info="http://www.rsyslog.com"] rsyslogd was HUPed**

**May 24 04:01:01 master systemd: Started Session 376 of user root.**

**May 24 04:01:01 master systemd: Starting Session 376 of user root.**

**May 24 05:01:01 master systemd: Started Session 377 of user root.**

**May 24 05:01:01 master systemd: Starting Session 377 of user root.**

**May 24 06:01:01 master systemd: Started Session 378 of user root.**

**May 24 06:01:01 master systemd: Starting Session 378 of user root.**

**May 24 07:01:01 master systemd: Started Session 379 of user root.**

**May 24 07:01:01 master systemd: Starting Session 379 of user root.**

**May 24 08:01:01 master systemd: Started Session 380 of user root.**

**May 24 08:01:01 master systemd: Starting Session 380 of user root.**

**May 24 09:01:01 master systemd: Started Session 381 of user root.**

**May 24 09:01:01 master systemd: Starting Session 381 of user root.**

**May 24 09:41:08 master systemd: Starting Cleanup of Temporary Directories...**

**May 24 09:41:08 master systemd: Started Cleanup of Temporary Directories.**

**May 24 10:01:01 master systemd: Started Session 382 of user root.**

**May 24 10:01:01 master systemd: Starting Session 382 of user root.**

**May 24 11:01:01 master systemd: Started Session 383 of user root.**

**May 24 11:01:01 master systemd: Starting Session 383 of user root.**

**May 24 12:01:01 master systemd: Started Session 384 of user root.**

**May 24 12:01:01 master systemd: Starting Session 384 of user root.**

**May 24 12:12:39 master su: (to root) root on pts/0**

**May 24 12:40:22 master su: (to hadoop) root on pts/0**

**May 24 12:40:42 master su: FAILED SU (to root) root on pts/0**

**May 24 12:40:51 master su: (to root) root on pts/0**

**May 24 13:01:01 master systemd: Started Session 385 of user root.**

**May 24 13:01:01 master systemd: Starting Session 385 of user root.**

**May 24 13:21:24 master systemd: Starting firewalld - dynamic firewall daemon...**

**May 24 13:21:26 master systemd: Started firewalld - dynamic firewall daemon.**

**May 24 13:21:26 master systemd: Reached target Network (Pre).**

**May 24 13:21:26 master systemd: Starting Network (Pre).**

**May 24 13:21:26 master kernel: ip6\_tables: (C) 2000-2006 Netfilter Core Team**

**May 24 13:21:26 master kernel: Ebtables v2.0 registered**

**May 24 13:21:27 master kernel: nf\_conntrack version 0.5.0 (65536 buckets, 262144 max)**

**May 24 13:21:27 master kernel: bridge: filtering via arp/ip/ip6tables is no longer available by default. Update your scripts to load br\_netfilter if you need this.**

**May 24 13:21:27 master kernel: Netfilter messages via NETLINK v0.30.**

**May 24 13:21:27 master kernel: ip\_set: protocol 6**

**May 24 13:21:27 master firewalld[104456]: WARNING: ICMP type 'beyond-scope' is not supported by the kernel for ipv6.**

**May 24 13:21:27 master firewalld[104456]: WARNING: beyond-scope: INVALID\_ICMPTYPE: No supported ICMP type., ignoring for run-time.**

**May 24 13:21:27 master firewalld[104456]: WARNING: ICMP type 'failed-policy' is not supported by the kernel for ipv6.**

**May 24 13:21:27 master firewalld[104456]: WARNING: failed-policy: INVALID\_ICMPTYPE: No supported ICMP type., ignoring for run-time.**

**May 24 13:21:27 master firewalld[104456]: WARNING: ICMP type 'reject-route' is not supported by the kernel for ipv6.**

**May 24 13:21:27 master firewalld[104456]: WARNING: reject-route: INVALID\_ICMPTYPE: No supported ICMP type., ignoring for run-time.**

**May 24 13:21:40 master systemd: Stopping firewalld - dynamic firewall daemon...**

**May 24 13:21:41 master kernel: Ebtables v2.0 unregistered**

**May 24 13:21:42 master systemd: Stopped firewalld - dynamic firewall daemon.**

**May 24 14:01:01 master systemd: Started Session 386 of user root.**

**May 24 14:01:01 master systemd: Starting Session 386 of user root.**

**May 24 15:01:01 master systemd: Started Session 387 of user root.**

**May 24 15:01:01 master systemd: Starting Session 387 of user root.**

**May 24 15:09:54 master su: (to hadoop) root on pts/0**

**May 24 15:24:53 master su: (to root) root on pts/0**

以上结果不仅包含了master主机用户切换的日志，而且包含了服务的状态，如防火墙服务状态的记录：“Started firewalld - dynamic firewall daemon”和“Stopped firewalld - dynamic firewall daemon.”。

#### 步骤二：查看计划任务日志/var/log/cron。

该文件会记录crontab计划任务的创建、执行信息。执行cat cron命令，显示如下：

[root@master log]# cat cron

**May 24 03:17:01 master run-parts(/etc/cron.daily)[88520]: finished logrotate**

**May 24 03:17:01 master run-parts(/etc/cron.daily)[88508]: starting man-db.cron**

**May 24 03:17:02 master run-parts(/etc/cron.daily)[88533]: finished man-db.cron**

**May 24 03:17:02 master anacron[88090]: Job `cron.daily' terminated**

**May 24 03:17:02 master anacron[88090]: Normal exit (1 job run)**

**May 24 04:01:01 master CROND[89710]: (root) CMD (run-parts /etc/cron.hourly)**

**May 24 04:01:01 master run-parts(/etc/cron.hourly)[89710]: starting 0anacron**

**May 24 04:01:01 master run-parts(/etc/cron.hourly)[89719]: finished 0anacron**

**May 24 05:01:01 master CROND[91290]: (root) CMD (run-parts /etc/cron.hourly)**

**May 24 05:01:01 master run-parts(/etc/cron.hourly)[91290]: starting 0anacron**

**May 24 05:01:01 master run-parts(/etc/cron.hourly)[91299]: finished 0anacron**

**May 24 06:01:01 master CROND[92855]: (root) CMD (run-parts /etc/cron.hourly)**

**May 24 06:01:01 master run-parts(/etc/cron.hourly)[92855]: starting 0anacron**

**May 24 06:01:01 master run-parts(/etc/cron.hourly)[92864]: finished 0anacron**

**May 24 07:01:01 master CROND[94414]: (root) CMD (run-parts /etc/cron.hourly)**

**May 24 07:01:01 master run-parts(/etc/cron.hourly)[94414]: starting 0anacron**

**May 24 07:01:01 master run-parts(/etc/cron.hourly)[94423]: finished 0anacron**

**May 24 08:01:01 master CROND[95975]: (root) CMD (run-parts /etc/cron.hourly)**

**May 24 08:01:01 master run-parts(/etc/cron.hourly)[95975]: starting 0anacron**

**May 24 08:01:01 master run-parts(/etc/cron.hourly)[95984]: finished 0anacron**

**May 24 09:01:01 master CROND[97536]: (root) CMD (run-parts /etc/cron.hourly)**

**May 24 09:01:01 master run-parts(/etc/cron.hourly)[97536]: starting 0anacron**

**May 24 09:01:01 master run-parts(/etc/cron.hourly)[97545]: finished 0anacron**

**May 24 10:01:01 master CROND[99143]: (root) CMD (run-parts /etc/cron.hourly)**

**May 24 10:01:01 master run-parts(/etc/cron.hourly)[99143]: starting 0anacron**

**May 24 10:01:01 master run-parts(/etc/cron.hourly)[99152]: finished 0anacron**

**May 24 11:01:01 master CROND[100743]: (root) CMD (run-parts /etc/cron.hourly)**

**May 24 11:01:01 master run-parts(/etc/cron.hourly)[100743]: starting 0anacron**

**May 24 11:01:01 master run-parts(/etc/cron.hourly)[100752]: finished 0anacron**

**May 24 12:01:01 master CROND[102303]: (root) CMD (run-parts /etc/cron.hourly)**

**May 24 12:01:01 master run-parts(/etc/cron.hourly)[102303]: starting 0anacron**

**May 24 12:01:01 master run-parts(/etc/cron.hourly)[102312]: finished 0anacron**

**May 24 13:01:01 master CROND[103904]: (root) CMD (run-parts /etc/cron.hourly)**

**May 24 13:01:01 master run-parts(/etc/cron.hourly)[103904]: starting 0anacron**

**May 24 13:01:01 master run-parts(/etc/cron.hourly)[103913]: finished 0anacron**

**May 24 14:01:01 master CROND[105657]: (root) CMD (run-parts /etc/cron.hourly)**

**May 24 14:01:01 master run-parts(/etc/cron.hourly)[105657]: starting 0anacron**

**May 24 14:01:01 master run-parts(/etc/cron.hourly)[105666]: finished 0anacron**

**May 24 15:01:01 master CROND[107242]: (root) CMD (run-parts /etc/cron.hourly)**

**May 24 15:01:01 master run-parts(/etc/cron.hourly)[107242]: starting 0anacron**

**May 24 15:01:01 master run-parts(/etc/cron.hourly)[107251]: finished 0anacron**

#### 步骤三：查看系统引导日志/var/log/dmesg。

该文件记录硬件设备信息(device)属纯文本,也可以用dmesg命令查看。由于文件内容比较多，截取了一部分的内容，显示如下：

[root@master log]$ dmesg

**...**

**[ 52.361746] NET: Registered protocol family 40**

**[ 52.913901] IPv6: ADDRCONF(NETDEV\_UP): ens33: link is not ready**

**[ 52.929079] e1000: ens33 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: None**

**[ 53.850087] IPv6: ens33: IPv6 duplicate address fe80::6b63:dc78:878e:35f3 detected!**

**[ 54.276104] IPv6: ens33: IPv6 duplicate address fe80::2e35:1d99:a67d:6df9 detected!**

**[1224098.178835] sched: RT throttling activated**

**[1310998.894561] ip6\_tables: (C) 2000-2006 Netfilter Core Team**

**[1310999.227491] Ebtables v2.0 registered**

**[1310999.403806] nf\_conntrack version 0.5.0 (65536 buckets, 262144 max)**

**[1310999.747928] bridge: filtering via arp/ip/ip6tables is no longer available by default. Update your scripts to load br\_netfilter if you need this.**

**[1310999.827082] Netfilter messages via NETLINK v0.30.**

**[1310999.840387] ip\_set: protocol 6**

**[1311014.282821] Ebtables v2.0 unregistered**

**...**

以上结果显示了网卡设备e1000启动，获取IPv6地址的过程。

#### 步骤四：查看邮件系统日志:/var/log/maillog。

该日志文件记录了每一个发送到系统或从系统发出的电子邮件的活动。它可以用来查看用户使用哪个系统发送工具或把数据发送到哪个系统。可以采用cat /var/log/maillog或者tail -f /var/log/maillog查看电子邮件的活动。

#### 步骤五：查看用户登录日志。

这种日志数据用于记录 Linux 操作系统用户登录及退出系统的相关信息，包括用户名、登录的终端、登录时间、来源主机、正在使用的进程操作等。

以下文件保存了用户登录、退出系统等相关信息

1）/var/log/lastlog ：最近的用户登录事件

2）/var/log/wtmp ：用户登录注销及系统开、关机事件

3）/var/run/utmp ：当前登录的每个用户的详细信息

4）/var/log/secure ：与用户验证相关的安全性事件

（1）lastlog 列出所有用户最近登录的信息

lastlog引用的是/var/log/lastlog文件中的信息，包括登录名、端口、最后登录时间等。

[root@master log]$ lastlog

**用户名 端口 来自 最后登陆时间**

**root pts/2 192.168.1.1 一 5月 18 16:46:11 +0800 2020**

**bin \*\*从未登录过\*\***

**daemon \*\*从未登录过\*\***

**adm \*\*从未登录过\*\***

**lp \*\*从未登录过\*\***

**sync \*\*从未登录过\*\***

**shutdown \*\*从未登录过\*\***

**halt \*\*从未登录过\*\***

**mail \*\*从未登录过\*\***

**operator \*\*从未登录过\*\***

**games \*\*从未登录过\*\***

**ftp \*\*从未登录过\*\***

**nobody \*\*从未登录过\*\***

**systemd-network \*\*从未登录过\*\***

**dbus \*\*从未登录过\*\***

**polkitd \*\*从未登录过\*\***

**postfix \*\*从未登录过\*\***

**sshd \*\*从未登录过\*\***

**chrony \*\*从未登录过\*\***

**hadoop pts/0 六 5月 23 18:03:27 +0800 2020**

**mysql \*\*从未登录过\*\***

**zhq pts/0 192.168.1.128 四 5月 7 17:56:45 +0800 2020**

（2）last 列出当前和曾经登入系统的用户信息

它默认读取的是/var/log/wtmp文件的信息。输出的内容包括：用户名、终端位置、登录源信息、开始时间、结束时间、持续时间。注意最后一行输出的是wtmp文件起始记录的时间。当然也可以通过last -f参数指定读取文件，可以是/var/log/btmp、/var/run/utmp文件。

执行命令last，

[root@master log]$ last

**root pts/2 192.168.1.1 Mon May 18 16:46 still logged in**

**root pts/1 192.168.1.1 Mon May 18 16:40 - 17:02 (00:21)**

**root pts/3 192.168.1.1 Mon May 18 16:39 - 16:40 (00:00)**

**root pts/2 192.168.1.1 Mon May 18 16:39 - 16:40 (00:00)**

**root pts/1 192.168.1.1 Mon May 18 16:38 - 16:40 (00:01)**

**root pts/1 192.168.1.1 Sat May 16 00:52 - 00:52 (00:00)**

**root tty1 Tue May 12 09:43 still logged in**

**hadoop tty1 Sat May 9 09:15 - 09:43 (3+00:27)**

**root pts/0 192.168.1.1 Sat May 9 09:13 still logged in**

**reboot system boot 3.10.0-693.el7.x Sat May 9 09:12 - 12:09 (15+02:57)**

**root pts/0 192.168.1.1 Thu May 7 22:06 - crash (1+11:05)**

**reboot system boot 3.10.0-693.el7.x Thu May 7 21:57 - 12:09 (16+14:12)**

**zhq pts/0 192.168.1.128 Thu May 7 17:56 - 18:42 (00:45)**

**root pts/2 192.168.1.1 Wed May 6 17:24 - down (1+04:31)**

**root pts/1 192.168.1.1 Wed May 6 17:24 - down (1+04:32)**

**root pts/0 192.168.1.1 Tue May 5 14:00 - 17:25 (1+03:24)**

**reboot system boot 3.10.0-693.el7.x Tue May 5 13:30 - 21:56 (2+08:26)**

**root pts/0 192.168.1.1 Mon May 4 21:36 - crash (15:54)**

**root pts/1 192.168.1.1 Mon May 4 21:19 - 21:36 (00:16)**

**root pts/0 192.168.1.1 Mon May 4 20:56 - 21:19 (00:23)**

**root tty1 Mon May 4 20:40 - 22:34 (01:53)**

**reboot system boot 3.10.0-693.el7.x Mon May 4 20:30 - 21:56 (3+01:26)**

**root pts/0 192.168.1.1 Thu Apr 23 15:47 - 16:14 (00:26)**

**root pts/0 192.168.1.1 Thu Apr 23 12:49 - 15:47 (02:58)**

**root pts/0 192.168.1.1 Thu Apr 23 11:10 - 11:19 (00:08)**

**hadoop pts/0 192.168.1.8 Thu Apr 23 11:09 - 11:09 (00:00)**

**hadoop pts/0 192.168.1.7 Thu Apr 23 11:09 - 11:09 (00:00)**

**root tty1 Thu Apr 23 11:11 - crash (11+09:18)**

**reboot system boot 3.10.0-693.el7.x Thu Apr 23 11:01 - 21:56 (14+10:55)**

**hadoop pts/0 ::1 Thu Apr 23 10:58 - 10:58 (00:00)**

**root tty1 Thu Apr 23 18:59 - 10:58 (-8:00)**

**reboot system boot 3.10.0-693.el7.x Thu Apr 23 18:58 - 10:58 (-7:-59)**

**root tty1 Thu Apr 23 18:58 - 18:58 (00:00)**

**reboot system boot 3.10.0-693.el7.x Thu Apr 23 18:57 - 10:58 (-7:-58)**

**wtmp begins Thu Apr 23 18:57:44 2020**

通过以上的结果显示，自4月23日以来，系统重启了7次，最近一次是root于5月18日16:46登录，现在仍在登录状态。

切换到root用户，使用命令last -f /var/run/utmp，查看utmp文件

[root@master log]# last -f /var/run/utmp

**root pts/2 192.168.1.1 Mon May 18 16:46 still logged in**

**root pts/0 192.168.1.1 Sat May 9 09:13 still logged in**

**root tty1 Tue May 12 09:43 still logg ed in**

**reboot system boot 3.10.0-693.el7.x Sat May 9 09:12 - 12:13 (15+03:01)**

**utmp begins Sat May 9 09:12:07 2020**

（3）lastb 列出失败尝试的登录信息

lastb和last命令功能完全相同，只不过它默认读取的是/var/log/btmp文件的信息。

[root@master log]# lastb

**btmp begins Mon May 4 21:35:50 2020**

上面结果显示，zhq和hadoop用户曾经登录失败。

（4）通过Linux 系统安全日志文件/var/log/secure 可查看SSH登录行为，该文件读取需要root权限。

切换为root用户，执行cat /var/log/secure 命令查看服务器登陆行为

[root@master log]# cat /var/log/secure

**May 24 12:12:39 master su: pam\_unix(su:session): session opened for user root by root(uid=1000)**

**May 24 12:40:22 master su: pam\_unix(su:session): session opened for user hadoop by root(uid=0)**

**May 24 12:40:39 master su: pam\_unix(su:auth): authentication failure; logname=root uid=1000 euid=0 tty=pts/0 ruser=hadoop rhost= user=root**

**May 24 12:40:39 master su: pam\_succeed\_if(su:auth): requirement "uid >= 1000" not met by user "root"**

**May 24 12:40:51 master su: pam\_unix(su:session): session opened for user root by root(uid=1000)**

### 实验任务二： 在Hadoop MapReduce Jobs中查看日志信息

首先，我们需要在MapReduce作业中记录信息，比如使用标准库，log4j和用System.out.println()或者System.err.println()写入标准输出流。Hadoop提供了一个查看日志的用户界面。

Hadoop中每一个Mapper和Reducer都有以下三种类型的日志：

（1）stdout-System.out.println()的输出定向到这个文件。

（2）stderr-System.err.println()的输出定向到这个文件。

（3）syslog-log4j的日记输出定向到这个文件。在作业执行中出现和没有被处理的所有异常的栈跟踪信息会在syslog中显示。

在浏览器地址栏中输入http://master:19888/jobhistory,将显示关于作业的摘要信息，如图15-1所示。

注意:需先启动jobhistory进程

启动： 在hadoop用户下执行

[hadoop@master ~]$ cd /usr/local/src/hadoop/sbin

[hadoop@master sbin]$ ./mr-jobhistory-daemon.sh start historyserver

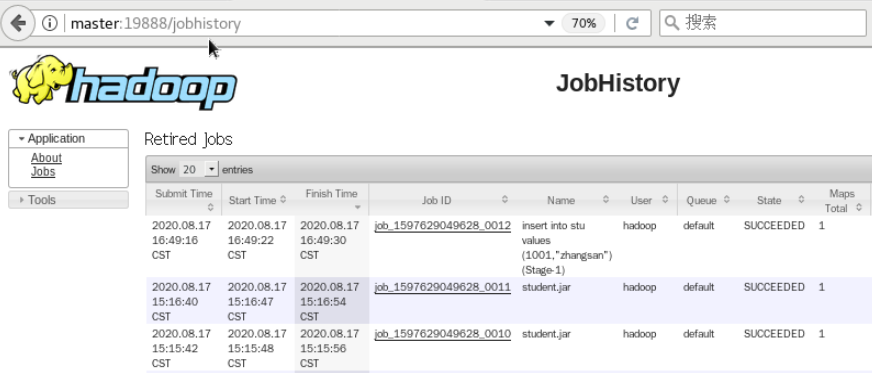


图15-1 作业历史记录查看器

单击Job ID为job\_1597629049628\_0012链接，会出现以下界面，如图15-2所示。

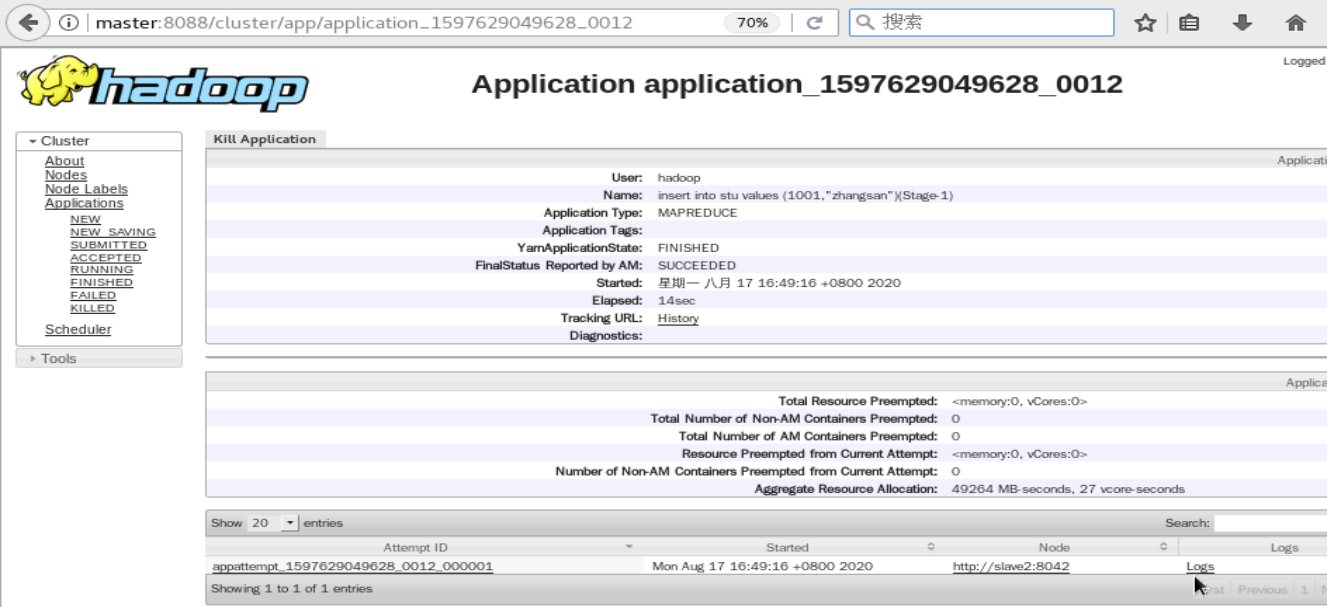


图15-2 作业详细视图

图15-2显示有关作业的额外信息，包括它的执行状态、开始和停止次数，以及运行所在的队列等基本信息。我们可以查看有多少Mapper和Reducer用于执行作业。注意Map、Shuffle、Sort、Reduce每个阶段的平均执行时间。现在我们可以点击链接去查看Mapper和Reducer的信息信息。

点击Maps的“1”的链接查看详细的Mapper日志，如图15-3所示。

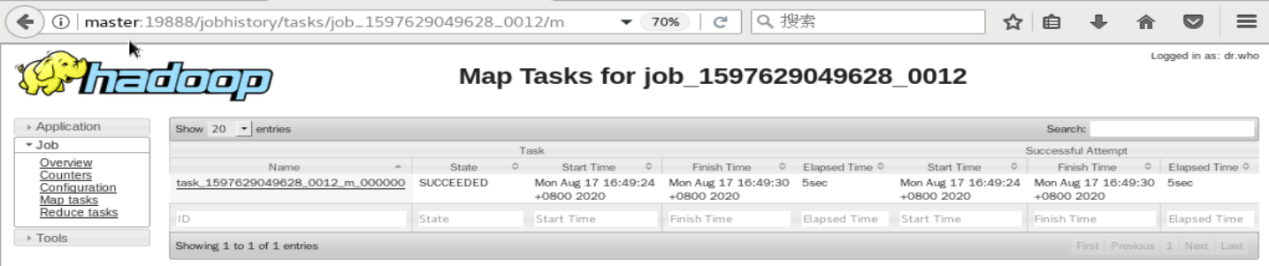


图15-3 Map详细视图

现在可以查看Mapper特定实例的日志。我们可以点击Logs（日志）以查看如下图15-4所示的信息。



图15-4 Mapper日志视图

从上图我们可以查看三种日志：

(1)stdout，标准输出；

(2)stderr，标准错误；

(3)syslog，系统日志。

如果不启用日志聚合，则会显示一条日志不可用的消息，如图15-5所示。

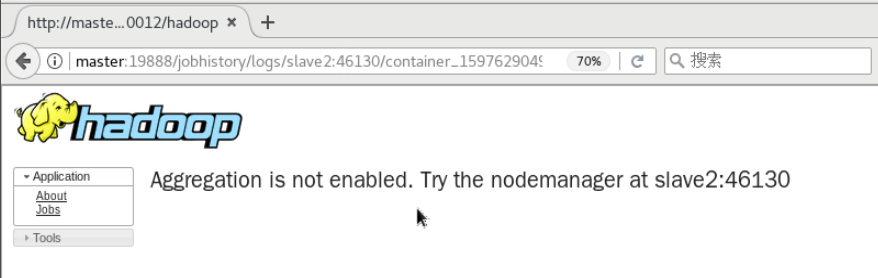


图15-5 Mapper 错误信息

在yarn-site.xml 中加入以下配置启动日志聚合

[hadoop@master ~]$ cd /usr/local/src/hadoop/etc/hadoop

[hadoop@master hadoop]$ vi yarn-site.xml

<property>

<name>yarn.log-aggregation-enable</name>

<value>true</value>

</property>

### 实验任务三：通过用户界面查看Hadoop日志

默认情况下，可以通过以下URL访问日志，http: //master：19888。

如前所述，日志是否聚集对用户都是透明的。如果日志是聚集的，Job History Manager将会把日志从HDFS中取回。如果日志是非聚集的，将通过向单个节点的节点管理器发送请求来获取日志。

作业运行的时候，能通过 Application Master Web界面查看的日志将可通过节点管理器web界面查看。 Application Master Web界面反过来可以通过资源管理器Web界面左边“RUNNING”的作业链接来访问。默认资源管理器Web界面可通过以下URL访问：http://master:8088如图15-6所示

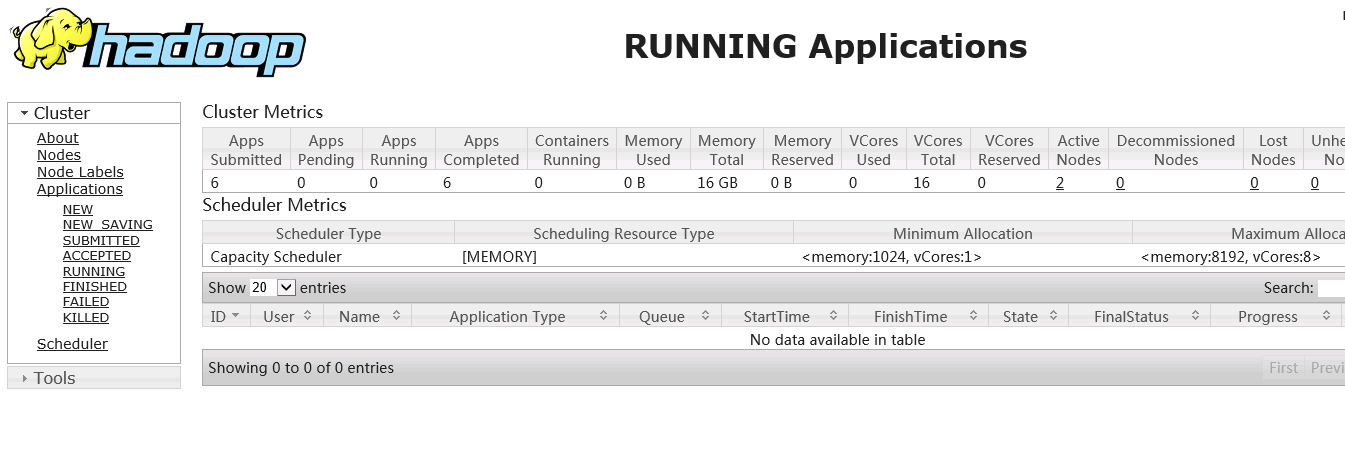


图15-6 查看正在运行中的作业信息

根据上图所示，当前未有运行中的作业，故显示正在运行的作业为空。

点击左边的菜单“FINISHED”显示已经完成运行的作业，如图15-7所示。

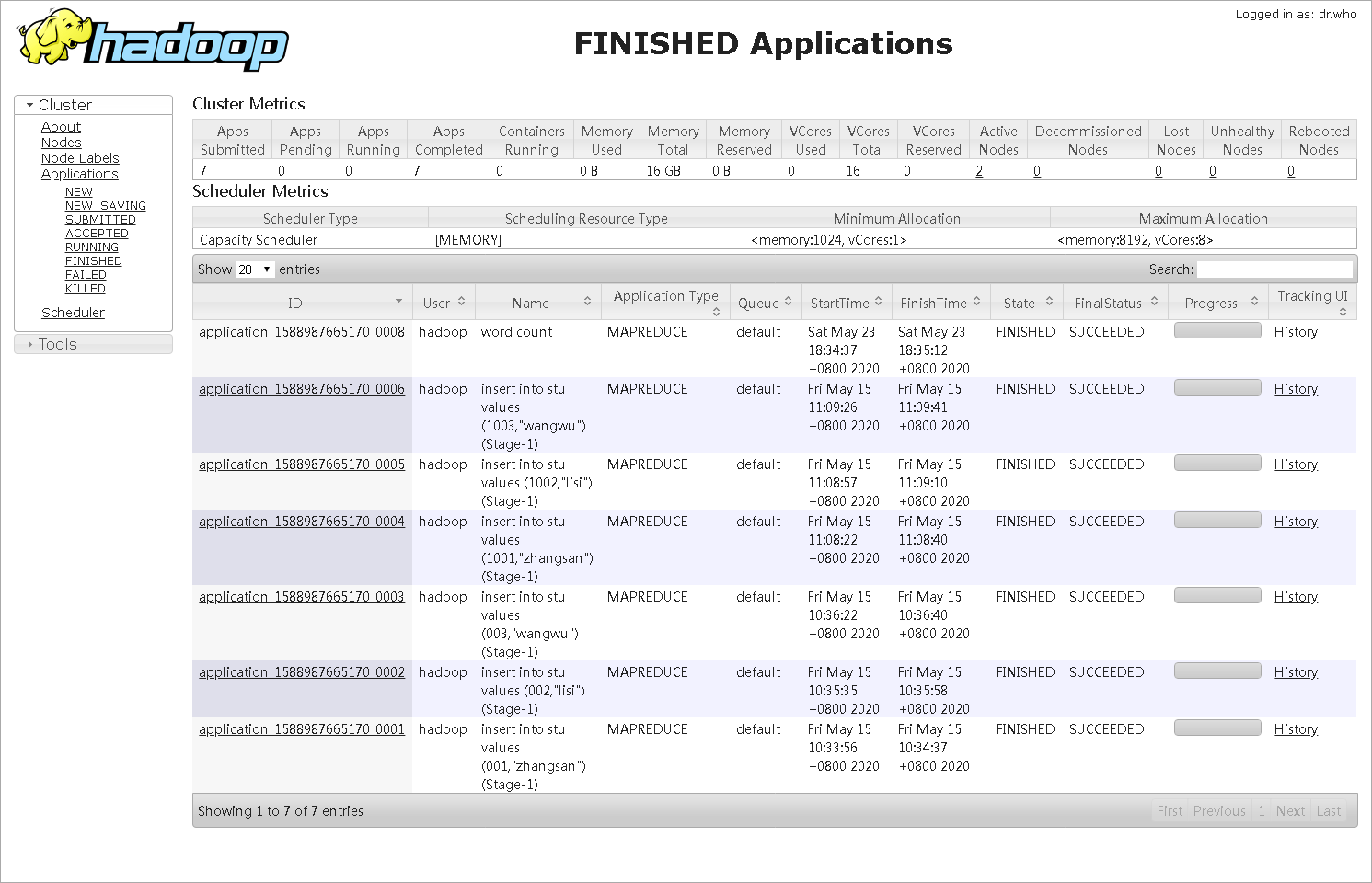


图15-7 完成作业列表

从上图，我们可以看出已完成的作业有7个，并且最终的状态为“SUCCEEDED（成功）”。我们选择ID号为“application\_1588987665170\_008”,点击“History（历史）”，可以查看该作业详细的日志信息，如图15-8所示。

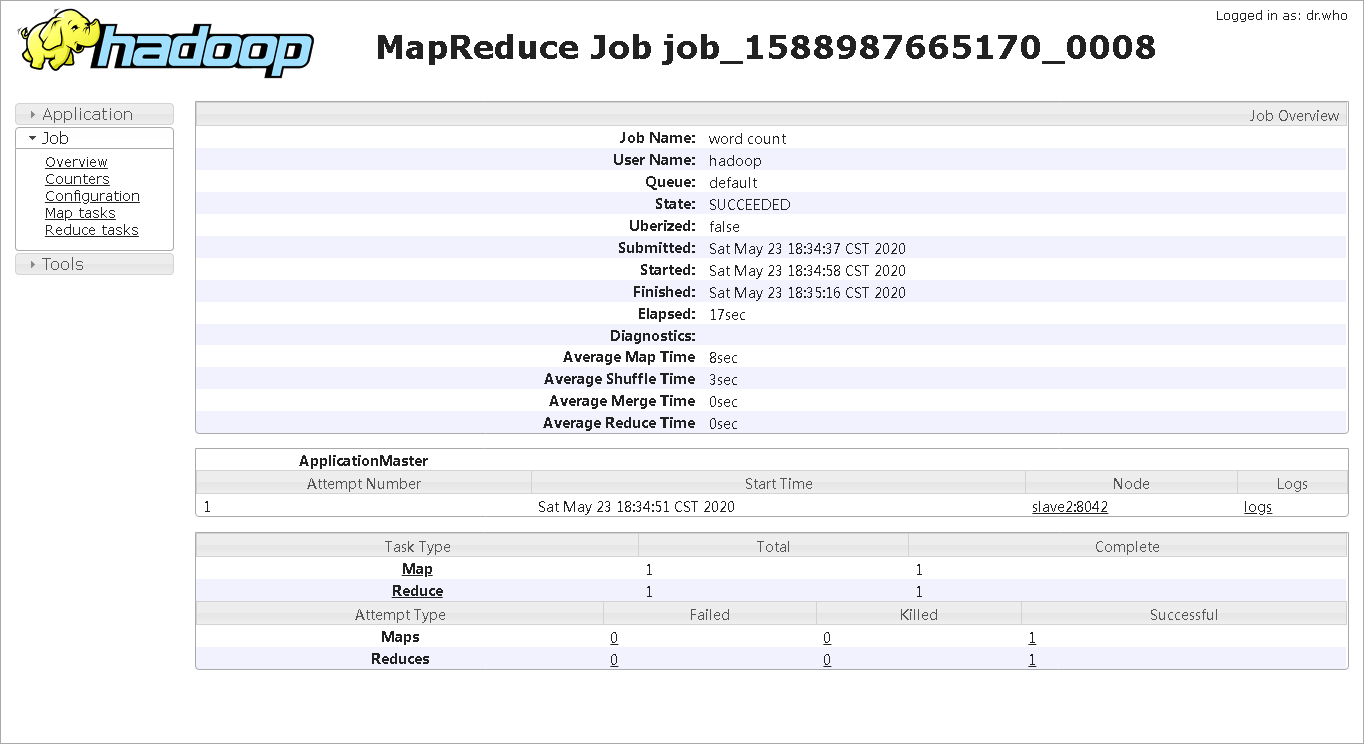


图15-8 作业日志信息

从上图可以了解到该作业信息如表15-2所示。

表15-2 作业job\_1588987665170\_008的日志主要信息

|  |  |  |
| --- | --- | --- |
| 序号 | 选项 | 值 |
| 1 | Job Name（作业名） | Word count |
| 2 | User Name（用户名） | Hadoop |
| 3 | Queue（队列） | False（默认） |
| 4 | State（状态） | SUCCEEDED（成功） |
| 5 | Submitted（提交时间） | Sat May 23 18:34:37 |
| 6 | Started（开始时间） | Sat May 23 18:34:58 |
| 7 | Finished（结束时间） | Sat May 23 18:35:16 |
| 8 | Elapsed（持续时间） | 17sec |
| 9 | Average Map Time | 8sec |
| 10 | Average Shuffle Time | 3sec |
| 11 | Average Merge Time | 0sec |
| 12 | Average Reduce Time | 0sec |

我们也可以通过Hadoop的用户界面查看日志信息，使用浏览器访问http://master:50070，点击Utilities-->Logs，如图,15-9 所示。

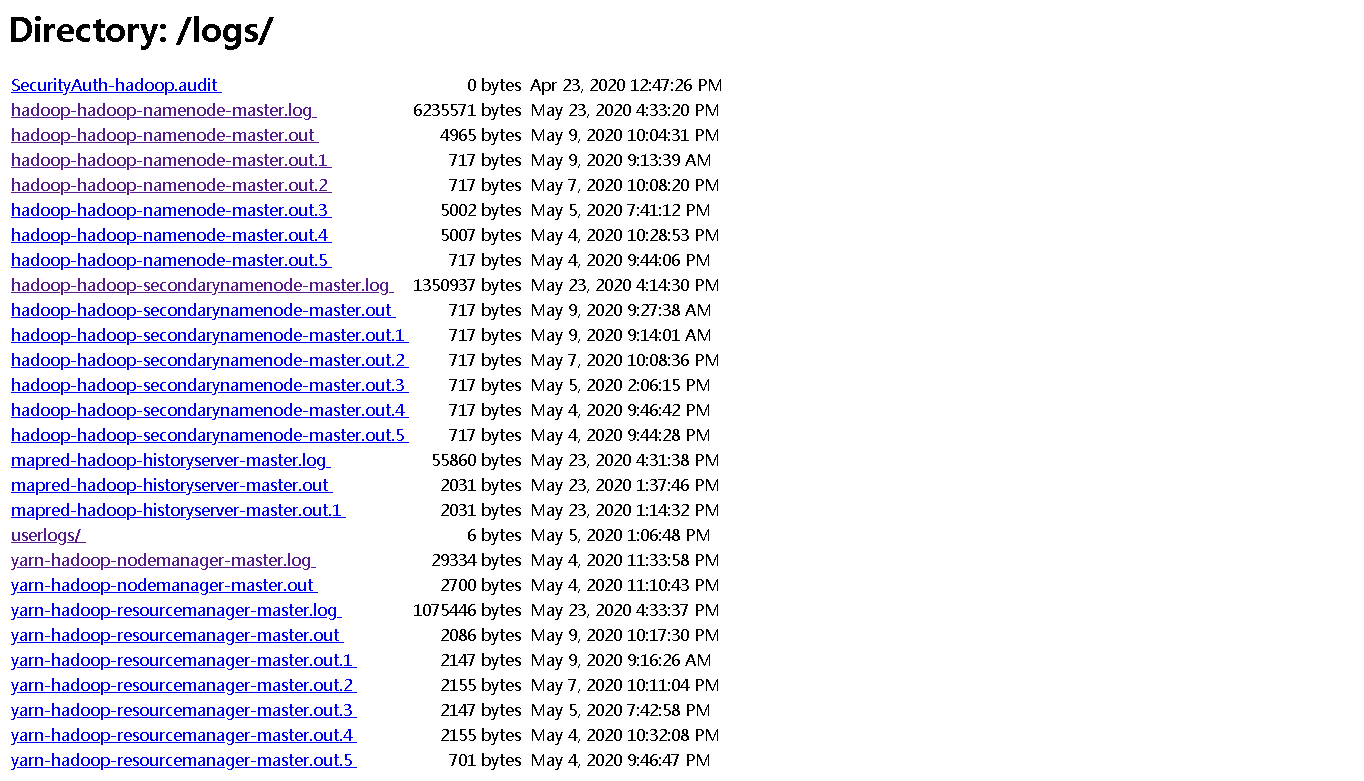


图15-9 显示Hadoop的日志文件

从上图，我们可以看到Hadoop中的日志文件列表，包括NameNode（名称节点）、SecondaryNamenode（二级名称节点）、HistoryServer（历史服务器）、NodeManager（节点管理器）和ResourceManager（资源管理器）等日志文件。点击相应的日志文件即可查看日志的内容。如点击查看hadoop-hadoop-namenode-master.log日志文件，如图15-10所示。



图15-10 查看hadoop-hadoop-namenode-master.log日志内容

### 实验任务四：通过命令查看Hadoop日志

可以通过与命令行交互的的方式获取Hadoop的日志文件列表。

当某个日志达到一定的大小，将会被切割出一个新的文件，切割出来的日志文件名类似 “XXX.log.数字”的形式，后面的数字越大，代表日志越旧。在默认情况下，只保存前 20 个日志文件。

使用hadoop用户登录，并切换到/usr/local/src/hadoop/logs目录，执行ll命令，查看日志列表。

[hadoop@master ~]$cd /usr/local/src/hadoop/logs

[hadoop@master logs]$ ll

**总用量 8652**

**-rw-rw-r-- 1 hadoop hadoop 6233910 5月 23 16:14 hadoop-hadoop-namenode-master.log**

**-rw-rw-r-- 1 hadoop hadoop 4965 5月 9 22:04 hadoop-hadoop-namenode-master.out**

**-rw-rw-r-- 1 hadoop hadoop 717 5月 9 09:13 hadoop-hadoop-namenode-master.out.1**

**-rw-rw-r-- 1 hadoop hadoop 717 5月 7 22:08 hadoop-hadoop-namenode-master.out.2**

**-rw-rw-r-- 1 hadoop hadoop 5002 5月 5 19:41 hadoop-hadoop-namenode-master.out.3**

**-rw-rw-r-- 1 hadoop hadoop 5007 5月 4 22:28 hadoop-hadoop-namenode-master.out.4**

**-rw-rw-r-- 1 hadoop hadoop 717 5月 4 21:44 hadoop-hadoop-namenode-master.out.5**

**-rw-rw-r-- 1 hadoop hadoop 1350937 5月 23 16:14 hadoop-hadoop-secondarynamenode-master.log**

**-rw-rw-r-- 1 hadoop hadoop 717 5月 9 09:27 hadoop-hadoop-secondarynamenode-master.out**

**-rw-rw-r-- 1 hadoop hadoop 717 5月 9 09:14 hadoop-hadoop-secondarynamenode-master.out.1**

**-rw-rw-r-- 1 hadoop hadoop 717 5月 7 22:08 hadoop-hadoop-secondarynamenode-master.out.2**

**-rw-rw-r-- 1 hadoop hadoop 717 5月 5 14:06 hadoop-hadoop-secondarynamenode-master.out.3**

**-rw-rw-r-- 1 hadoop hadoop 717 5月 4 21:46 hadoop-hadoop-secondarynamenode-master.out.4**

**-rw-rw-r-- 1 hadoop hadoop 717 5月 4 21:44 hadoop-hadoop-secondarynamenode-master.out.5**

**-rw-rw-r-- 1 hadoop hadoop 55497 5月 23 16:22 mapred-hadoop-historyserver-master.log**

**-rw-rw-r-- 1 hadoop hadoop 2031 5月 23 13:37 mapred-hadoop-historyserver-master.out**

**-rw-rw-r-- 1 hadoop hadoop 2031 5月 23 13:14 mapred-hadoop-historyserver-master.out.1**

**-rw-rw-r-- 1 hadoop hadoop 0 4月 23 12:47 SecurityAuth-hadoop.audit**

**drwxr-xr-x 2 hadoop hadoop 6 5月 5 13:06 userlogs**

**-rw-rw-r-- 1 hadoop hadoop 29334 5月 4 23:33 yarn-hadoop-nodemanager-master.log**

**-rw-rw-r-- 1 hadoop hadoop 2700 5月 4 23:10 yarn-hadoop-nodemanager-master.out**

**-rw-rw-r-- 1 hadoop hadoop 1075192 5月 23 15:52 yarn-hadoop-resourcemanager-master.log**

**-rw-rw-r-- 1 hadoop hadoop 2086 5月 9 22:17 yarn-hadoop-resourcemanager-master.out**

**-rw-rw-r-- 1 hadoop hadoop 2147 5月 9 09:16 yarn-hadoop-resourcemanager-master.out.1**

**-rw-rw-r-- 1 hadoop hadoop 2155 5月 7 22:11 yarn-hadoop-resourcemanager-master.out.2**

**-rw-rw-r-- 1 hadoop hadoop 2147 5月 5 19:42 yarn-hadoop-resourcemanager-master.out.3**

**-rw-rw-r-- 1 hadoop hadoop 2155 5月 4 22:32 yarn-hadoop-resourcemanager-master.out.4**

**-rw-rw-r-- 1 hadoop hadoop 701 5月 4 21:46 yarn-hadoop-resourcemanager-master.out.5**

我们可以获知日志文件的大小和Hadoop中所属组件的日记文件；yarn-hadoop-resourcemanager-master.out文件被切割为五个文件，并且后面的数字越大，代表该文件越旧，符合Hadoop日志文件切割原则。

### 实验任务五：查看HBase日志

Hbase提供了Web用户界面对日志文件的查看，使用浏览器访问http://master:60010，显示HBase的web主界面，如图15-11所示。

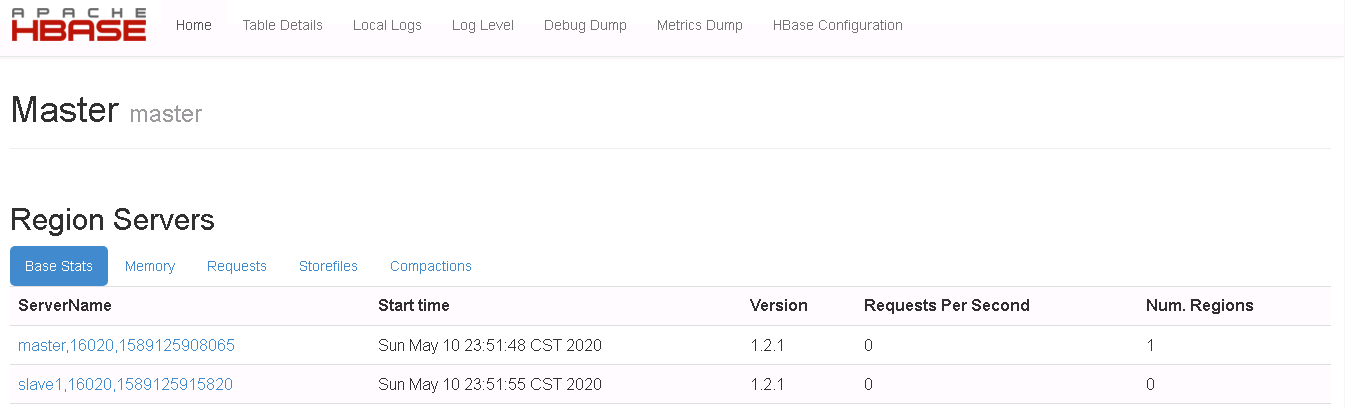


图15-11 HBase web用户界面

点击“Local Logs”菜单打开HBase的日志列表，如图15-12所示。

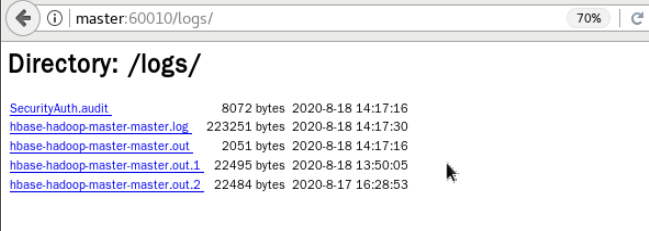


图15-12 HBase日志列表

点击其中一条链接来访问相应的日志信息，如hbase-hadoop-master-master.log日志内容如图15-13 所示。

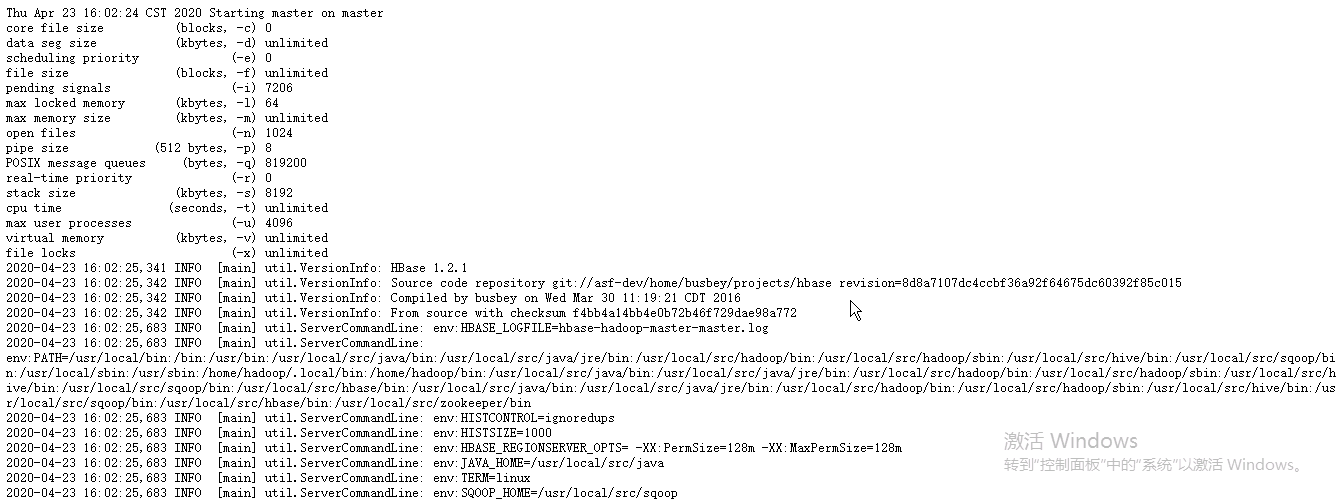


图15-13 hbase-hadoop-master-master.log日志内容（部份）

### 实验任务六：查看Hive日志

Hive日志存储的位置为/tmp/hadoop，在命令行的模式下，切换到该目录，执行ll命令，查看Hive的日志列表，显示如下。

[root@master hadoop]# cd /tmp/hadoop

[root@master hadoop]# ll

**总用量 76**

**-rw-rw-r-- 1 hadoop hadoop 1968 5月 24 20:43 hive.log**

**-rw-rw-r-- 1 hadoop hadoop 984 5月 15 23:49 hive.log.2020-05-12**

**-rw-rw-r-- 1 hadoop hadoop 984 5月 15 23:49 hive.log.2020-05-13**

**-rw-rw-r-- 1 hadoop hadoop 738 5月 16 18:26 hive.log.2020-05-15**

**-rw-rw-r-- 1 hadoop hadoop 1230 5月 18 17:44 hive.log.2020-05-16**

**-rw-rw-r-- 1 hadoop hadoop 2952 5月 23 18:36 hive.log.2020-05-18**

**-rw-rw-r-- 1 hadoop hadoop 4704 5月 23 18:41 hive.log.2020-05-23**

**-rw-rw-r-- 1 hadoop hadoop 42843 5月 23 18:41 stderr**

使用cat命令查看hive.log日志文件，显示如下：

[root@master hadoop]# cat hive.log

**2020-05-24T09:28:45,688 INFO [org.apache.hadoop.hive.common.JvmPauseMonitor$Monitor@37d3e140]: common.JvmPauseMonitor (JvmPauseMonitor.java:run(194)) - Detected pause in JVM or host machine (eg GC): pause of approximately 4912ms**

**No GCs detected**

**2020-05-24T09:28:45,689 INFO [org.apache.hadoop.hive.common.JvmPauseMonitor$Monitor@31a848ec]: common.JvmPauseMonitor (JvmPauseMonitor.java:run(194)) - Detected pause in JVM or host machine (eg GC): pause of approximately 4759ms**

**No GCs detected**

**2020-05-24T15:07:48,891 INFO [org.apache.hadoop.hive.common.JvmPauseMonitor$Monitor@37d3e140]: common.JvmPauseMonitor (JvmPauseMonitor.java:run(194)) - Detected pause in JVM or host machine (eg GC): pause of approximately 4916ms**

**No GCs detected**

**2020-05-24T15:07:48,892 INFO [org.apache.hadoop.hive.common.JvmPauseMonitor$Monitor@31a848ec]: common.JvmPauseMonitor (JvmPauseMonitor.java:run(194)) - Detected pause in JVM or host machine (eg GC): pause of approximately 3911ms**

**No GCs detected**

**2020-05-24T15:10:17,026 INFO [org.apache.hadoop.hive.common.JvmPauseMonitor$Monitor@31a848ec]: common.JvmPauseMonitor (JvmPauseMonitor.java:run(194)) - Detected pause in JVM or host machine (eg GC): pause of approximately 3161ms**

**No GCs detected**

**2020-05-24T15:10:17,026 INFO [org.apache.hadoop.hive.common.JvmPauseMonitor$Monitor@37d3e140]: common.JvmPauseMonitor (JvmPauseMonitor.java:run(194)) - Detected pause in JVM or host machine (eg GC): pause of approximately 4760ms**

**No GCs detected**

**2020-05-24T20:43:09,023 INFO [org.apache.hadoop.hive.common.JvmPauseMonitor$Monitor@37d3e140]: common.JvmPauseMonitor (JvmPauseMonitor.java:run(194)) - Detected pause in JVM or host machine (eg GC): pause of approximately 4767ms**

**No GCs detected**

**2020-05-24T20:43:09,016 INFO [org.apache.hadoop.hive.common.JvmPauseMonitor$Monitor@31a848ec]: common.JvmPauseMonitor (JvmPauseMonitor.java:run(194)) - Detected pause in JVM or host machine (eg GC): pause of approximately 4819ms**

**No GCs detected**

# 实验二 查看大数据平台告警信息

## 实验目的

完成本实验，您应该能够：

* 掌握大数据平台日志和告警的查看
* 掌握常见告警问题的解决方法

## 实验要求

* 了解大数据平台日志信息
* 了解大数据平台告警信息

## 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 |
| 大数据平台 | H3C DataEngine-E0104-RHEL6-X86\_64 |
| 服务和组件 | 完成前面章节的实验，其他服务及组件根据实验需求安装 |

## 实验过程

### 实验任务一：查看大数据平台主机告警信息

主机是大数据平台重要的基础设施，包含硬件资源（CPU、内存、存储等）和操作系统（Linux），而Linux操作系统管理着硬件资源，按需求调度CPU、内存和存储等资源，通过Linux操作查看相关日志的告警信息，可以了解硬件资源的状态，从而帮助运维人员快速定位问题，解决问题。

Linux操作系统的的日志文件存储在/var/log文件夹中。我们可以利用日志管理工具journalctl查看Linux操作系统主机上的告警信息。journalctl是centos7上专有的日志管理工具，该工具是从message这个文件里读取信息。

切换到/var/log文件夹，执行命令journalctl -p err..alert查询系统错误告警信息，显示如下：

[root@master log]# journalctl -p err..alert

**-- Logs begin at 六 2020-05-09 09:11:55 CST, end at 一 2020-05-25 13:01:01 CST.**

**5月 09 09:11:57 localhost.localdomain kernel: sd 2:0:0:0: [sda] Assuming drive c**

**5月 09 09:12:11 master kernel: piix4\_smbus 0000:00:07.3: SMBus Host Controller n**

**5月 09 09:12:20 master systemd[1]: Failed to start Postfix Mail Transport Agent.**

**5月 12 09:52:41 master sftp-server[37883]: error: Unknown extended request "fs-m**

**5月 12 09:52:41 master sftp-server[37883]: error: Unknown extended request "vend**

**5月 12 14:59:16 master sftp-server[44284]: error: Unknown extended request "fs-m**

**5月 12 14:59:16 master sftp-server[44284]: error: Unknown extended request "vend**

**5月 12 15:31:16 master sftp-server[46010]: error: Unknown extended request "fs-m**

**5月 12 15:31:16 master sftp-server[46010]: error: Unknown extended request "vend**

**5月 18 16:39:18 master sshd[13061]: error: Received disconnect from 192.168.1.1**

**5月 18 16:39:24 master sshd[13067]: error: Received disconnect from 192.168.1.1**

结果显示了2020年5月9日09:11:55到2020年5月25日13:01:01之间的错误告警信息。其中关于SysTemd、Sftp-Server和SSHD等服务的错误信息分别有1条、6条和2条。例如，其中“ Failed to start Postfix Mail Transport Agent.”说明Postfix邮件传输代理无法启动。我们通过查看分析Linux操作系统主机的告警信息，就能有针对性的解决各种服务的问题。

我们也可以使用journalctl命令，根据服务的ID号来查询其告警信息。如根据上面的结果显示，我们得知sshd服务的ID为13067，查询SSHD服务错误告警信息，执行命令journalctl \_PID=13067 -p err，结果显示如下。

[root@master log]# journalctl \_PID=13067 -p err

**-- Logs begin at 六 2020-05-09 09:11:55 CST, end at 一 2020-05-25 15:01:01 CST.**

**5月 18 16:39:24 master sshd[13067]: error: Received disconnect from 192.168.1.1**

**lines 1-2/2 (END)**

结果显示，2020年5月18日16:39:24，SSHD服务产生一个错误告警信息：从192.168.1.1接收到一个断开的信息。

### 实验任务二：查看Hadoop告警信息

Hadoop的日志主要是存在/usr/local/src/hadoop/logs文件夹中，而日志文件包含Hadoop各组件的状态和告警信息。切换到/usr/local/src/hadoop/logs目录，文件列表如下：

[root@master hadoop]# cd /usr/local/src/hadoop/logs

[root@master logs]# ll

**总用量 9604**

**-rw-rw-r-- 1 hadoop hadoop 6883829 5月 25 18:14 hadoop-hadoop-namenode-master.log**

**-rw-rw-r-- 1 hadoop hadoop 4965 5月 9 22:04 hadoop-hadoop-namenode-master.out**

**-rw-rw-r-- 1 hadoop hadoop 717 5月 9 09:13 hadoop-hadoop-namenode-master.out.1**

**-rw-rw-r-- 1 hadoop hadoop 717 5月 7 22:08 hadoop-hadoop-namenode-master.out.2**

**-rw-rw-r-- 1 hadoop hadoop 5002 5月 5 19:41 hadoop-hadoop-namenode-master.out.3**

**-rw-rw-r-- 1 hadoop hadoop 5007 5月 4 22:28 hadoop-hadoop-namenode-master.out.4**

**-rw-rw-r-- 1 hadoop hadoop 717 5月 4 21:44 hadoop-hadoop-namenode-master.out.5**

**-rw-rw-r-- 1 hadoop hadoop 1458238 5月 25 18:14 hadoop-hadoop-secondarynamenode-master.log**

**-rw-rw-r-- 1 hadoop hadoop 717 5月 9 09:27 hadoop-hadoop-secondarynamenode-master.out**

**-rw-rw-r-- 1 hadoop hadoop 717 5月 9 09:14 hadoop-hadoop-secondarynamenode-master.out.1**

**-rw-rw-r-- 1 hadoop hadoop 717 5月 7 22:08 hadoop-hadoop-secondarynamenode-master.out.2**

**-rw-rw-r-- 1 hadoop hadoop 717 5月 5 14:06 hadoop-hadoop-secondarynamenode-master.out.3**

**-rw-rw-r-- 1 hadoop hadoop 717 5月 4 21:46 hadoop-hadoop-secondarynamenode-master.out.4**

**-rw-rw-r-- 1 hadoop hadoop 717 5月 4 21:44 hadoop-hadoop-secondarynamenode-master.out.5**

**-rw-rw-r-- 1 hadoop hadoop 222747 5月 25 18:28 mapred-hadoop-historyserver-master.log**

**-rw-rw-r-- 1 hadoop hadoop 2031 5月 23 18:28 mapred-hadoop-historyserver-master.out**

**-rw-rw-r-- 1 hadoop hadoop 115 5月 23 18:27 mapred-hadoop-historyserver-master.out.1**

**-rw-rw-r-- 1 hadoop hadoop 2031 5月 23 13:37 mapred-hadoop-historyserver-master.out.2**

**-rw-rw-r-- 1 hadoop hadoop 2031 5月 23 13:14 mapred-hadoop-historyserver-master.out.3**

**-rw-rw-r-- 1 hadoop hadoop 0 4月 23 12:47 SecurityAuth-hadoop.audit**

**drwxr-xr-x 2 hadoop hadoop 6 5月 5 13:06 userlogs**

**-rw-rw-r-- 1 hadoop hadoop 29334 5月 4 23:33 yarn-hadoop-nodemanager-master.log**

**-rw-rw-r-- 1 hadoop hadoop 2700 5月 4 23:10 yarn-hadoop-nodemanager-master.out**

**-rw-rw-r-- 1 hadoop hadoop 1112596 5月 25 09:42 yarn-hadoop-resourcemanager-master.log**

**-rw-rw-r-- 1 hadoop hadoop 2086 5月 9 22:17 yarn-hadoop-resourcemanager-master.out**

**-rw-rw-r-- 1 hadoop hadoop 2147 5月 9 09:16 yarn-hadoop-resourcemanager-master.out.1**

**-rw-rw-r-- 1 hadoop hadoop 2155 5月 7 22:11 yarn-hadoop-resourcemanager-master.out.2**

**-rw-rw-r-- 1 hadoop hadoop 2147 5月 5 19:42 yarn-hadoop-resourcemanager-master.out.3**

**-rw-rw-r-- 1 hadoop hadoop 2155 5月 4 22:32 yarn-hadoop-resourcemanager-master.out.4**

**-rw-rw-r-- 1 hadoop hadoop 701 5月 4 21:46 yarn-hadoop-resourcemanager-master.out.5**

我们通过查看某个日志文件中包含告警信息的行，然后将这些行显示出来，如查询ResourceManager日记最新1000行且包含“info”关键字的告警信息，执行命令tail -1000f yarn-hadoop-resourcemanager-master.log | grep info，结果显示如下。

[root@master logs]# tail -1000f yarn-hadoop-resourcemanager-master.log | grep info

**2020-05-15 10:33:56,833 INFO org.apache.hadoop.yarn.server.resourcemanager.recovery.RMStateStore: Storing info for app: application\_1588987665170\_0001**

**2020-05-15 10:34:37,657 INFO org.apache.hadoop.yarn.server.resourcemanager.recovery.RMStateStore: Updating info for app: application\_1588987665170\_0001**

**2020-05-15 10:35:35,984 INFO org.apache.hadoop.yarn.server.resourcemanager.recovery.RMStateStore: Storing info for app: application\_1588987665170\_0002**

**2020-05-15 10:35:58,660 INFO org.apache.hadoop.yarn.server.resourcemanager.recovery.RMStateStore: Updating info for app: application\_1588987665170\_0002**

**2020-05-15 10:36:22,103 INFO org.apache.hadoop.yarn.server.resourcemanager.recovery.RMStateStore: Storing info for app: application\_1588987665170\_0003**

**2020-05-15 10:36:40,809 INFO org.apache.hadoop.yarn.server.resourcemanager.recovery.RMStateStore: Updating info for app: application\_1588987665170\_0003**

**2020-05-15 11:08:22,505 INFO org.apache.hadoop.yarn.server.resourcemanager.recovery.RMStateStore: Storing info for app: application\_1588987665170\_0004**

**2020-05-15 11:08:40,694 INFO org.apache.hadoop.yarn.server.resourcemanager.recovery.RMStateStore: Updating info for app: application\_1588987665170\_0004**

**2020-05-15 11:08:57,301 INFO org.apache.hadoop.yarn.server.resourcemanager.recovery.RMStateStore: Storing info for app: application\_1588987665170\_0005**

**2020-05-15 11:09:10,752 INFO org.apache.hadoop.yarn.server.resourcemanager.recovery.RMStateStore: Updating info for app: application\_1588987665170\_0005**

**2020-05-15 11:09:26,177 INFO org.apache.hadoop.yarn.server.resourcemanager.recovery.RMStateStore: Storing info for app: application\_1588987665170\_0006**

**2020-05-15 11:09:41,358 INFO org.apache.hadoop.yarn.server.resourcemanager.recovery.RMStateStore: Updating info for app: application\_1588987665170\_0006**

**2020-05-23 18:34:37,343 INFO org.apache.hadoop.yarn.server.resourcemanager.recovery.RMStateStore: Storing info for app: application\_1588987665170\_0008**

**2020-05-23 18:35:12,957 INFO org.apache.hadoop.yarn.server.resourcemanager.recovery.RMStateStore: Updating info for app: application\_1588987665170\_0008**

### 实验任务三：查看HBase告警信息

#### 步骤一：变更日志告警级别

在HBase的Web 用户界面提供了日志告警级别的查询和设置功能。在浏览器中访问http://master:60010/logLevel页面，显示如图15-14所示。



图15-14 HBase Log Level页面

若要查询某个日志的告警级别，输入该日志名，点击“Get Log Level”按钮，显示该日志的告警级别。如查询日志文件habase-hadoop-master-master.log的告警级别，如图15-15所示。



图15-15 输入要查询告警级别的日志

点击“Get Log Level”按钮，结果如图15-16所示。



图15-16 查询日志告警级别结果

结果显示，日志文件habase-hadoop-master-master.log的告警级别为INFO。如果要将该日志告警级别调整为WARN，则在第二个框中输入Log：habase-hadoop-master-master.log，Level：WARN，点击“Set Log Level”按钮，结果如图15-17所示。

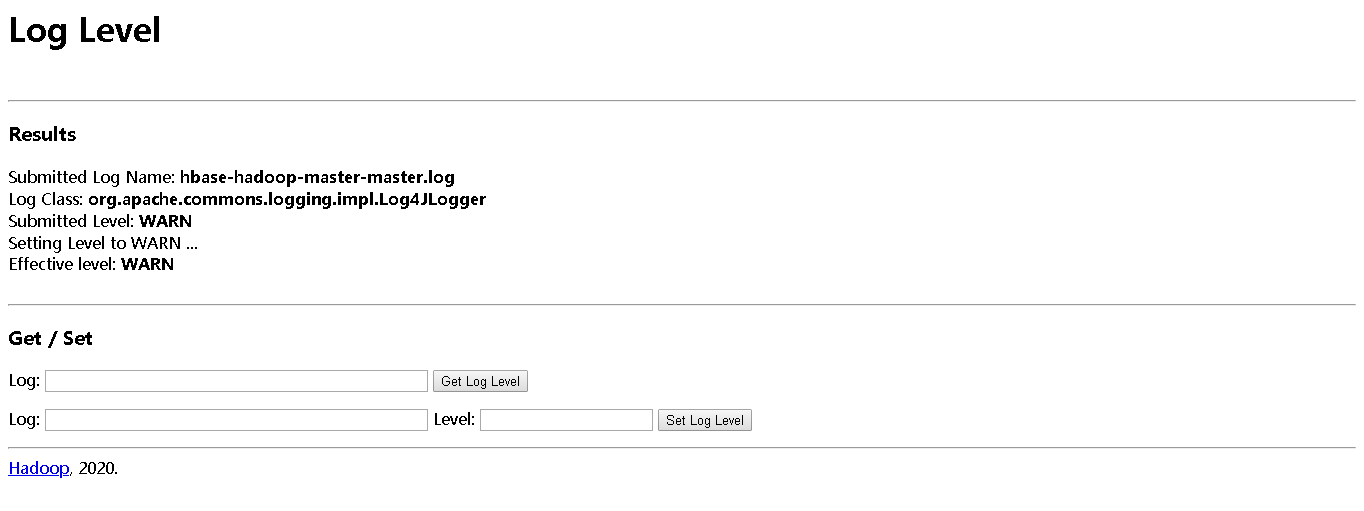


图15-17 日志告警级别变更

再次查询该日志文件的级别，结果如图15-18所示。



图15-18 查询日志告警级别

结果显示，habase-hadoop-master-master.log日志告警级别已变更为WARN级别。

#### 步骤二：查询日志告警信息

HBase的日志文件存储在/usr/loacl/src/hbase/logs目录中，切换到该目录下，查看 hbase-hadoop-master-master.log 文件的“INFO”告警信息， 执行命令tail -100f hbase-hadoop-master-master.log |grep INFO，结果显示如下。

[root@master hadoop]# cd /usr/local/src/hbase/logs

[root@master logs]# tail -100f hbase-hadoop-master-master.log |grep INFO

**2020-05-25 03:52:06,260 INFO [WALProcedureStoreSyncThread] wal.WALProcedureStore: Removed logs: [hdfs://master:9000/hbase/MasterProcWALs/state-00000000000000000485.log]**

**2020-05-25 04:52:06,277 INFO [WALProcedureStoreSyncThread] wal.WALProcedureStore: Remove log: hdfs://master:9000/hbase/MasterProcWALs/state-00000000000000000485.log**

**2020-05-25 04:52:06,277 INFO [WALProcedureStoreSyncThread] wal.WALProcedureStore: Removed logs: [hdfs://master:9000/hbase/MasterProcWALs/state-00000000000000000486.log]**

**2020-05-25 05:52:06,428 INFO [WALProcedureStoreSyncThread] wal.WALProcedureStore: Remove log: hdfs://master:9000/hbase/MasterProcWALs/state-00000000000000000486.log**

**2020-05-25 05:52:06,428 INFO [WALProcedureStoreSyncThread] wal.WALProcedureStore: Removed logs: [hdfs://master:9000/hbase/MasterProcWALs/state-00000000000000000487.log]**

**2020-05-25 06:52:06,445 INFO [WALProcedureStoreSyncThread] wal.WALProcedureStore: Remove log: hdfs://master:9000/hbase/MasterProcWALs/state-00000000000000000487.log**

**2020-05-25 06:52:06,445 INFO [WALProcedureStoreSyncThread] wal.WALProcedureStore: Removed logs: [hdfs://master:9000/hbase/MasterProcWALs/state-00000000000000000488.log]**

**2020-05-25 07:52:06,467 INFO [WALProcedureStoreSyncThread] wal.WALProcedureStore: Remove log: hdfs://master:9000/hbase/MasterProcWALs/state-00000000000000000488.log**

**2020-05-25 07:52:06,467 INFO [WALProcedureStoreSyncThread] wal.WALProcedureStore: Removed logs: [hdfs://master:9000/hbase/MasterProcWALs/state-00000000000000000489.log]**

**2020-05-25 08:52:06,487 INFO [WALProcedureStoreSyncThread] wal.WALProcedureStore: Remove log: hdfs://master:9000/hbase/MasterProcWALs/state-00000000000000000489.log**

**2020-05-25 08:52:06,487 INFO [WALProcedureStoreSyncThread] wal.WALProcedureStore: Removed logs: [hdfs://master:9000/hbase/MasterProcWALs/state-00000000000000000490.log]**

查看 hbase-hadoop-master-master.log 文件的“WARN”级别告警信息， 执行命令tail -100f hbase-hadoop-master-master.log |grep WARN，结果显示如下。

[root@master hadoop]# cd /usr/local/src/hbase/logs

[root@master logs]# tail -100f hbase-hadoop-master-master.log |grep WARN

**2020-05-25 18:54:54,388 WARN [master,16000,1589125905790\_ChoreService\_1] cleaner.TimeToLiveLogCleaner: Found a log (hdfs://master:9000/hbase/oldWALs/slave2%2C16020%2C1589125917741.default.1590400496064) newer than current time (1590404094387 < 1590404096420), probably a clock skew**

**2020-05-25 21:54:54,566 WARN [master,16000,1589125905790\_ChoreService\_1] cleaner.TimeToLiveLogCleaner: Found a log (hdfs://master:9000/hbase/oldWALs/slave1%2C16020%2C1589125915820.default.1590411294823) newer than current time (1590414894566 < 1590414895141), probably a clock skew**

**2020-05-25 21:54:54,567 WARN [master,16000,1589125905790\_ChoreService\_1] cleaner.TimeToLiveLogCleaner: Found a log (hdfs://master:9000/hbase/oldWALs/slave2%2C16020%2C1589125917741.default.1590411296989) newer than current time (1590414894567 < 1590414897324), probably a clock skew**

**2020-05-25 22:54:55,097 WARN [master,16000,1589125905790\_ChoreService\_1] cleaner.TimeToLiveLogCleaner: Found a log (hdfs://master:9000/hbase/oldWALs/slave1%2C16020%2C1589125915820.default.1590414895100) newer than current time (1590418495097 < 1590418495409), probably a clock skew**

**2020-05-25 22:54:55,097 WARN [master,16000,1589125905790\_ChoreService\_1] cleaner.TimeToLiveLogCleaner: Found a log (hdfs://master:9000/hbase/oldWALs/slave2%2C16020%2C1589125917741.default.1590414897294) newer than current time (1590418495097 < 1590418497591), probably a clock skew**

**2020-05-25 23:54:57,182 WARN [master,16000,1589125905790\_ChoreService\_1] cleaner.TimeToLiveLogCleaner: Found a log (hdfs://master:9000/hbase/oldWALs/slave2%2C16020%2C1589125917741.default.1590418497558) newer than current time (1590422097182 < 1590422097857), probably a clock skew**

**2020-05-26 00:54:54,407 WARN [master,16000,1589125905790\_ChoreService\_1] cleaner.TimeToLiveLogCleaner: Found a log (hdfs://master:9000/hbase/oldWALs/slave1%2C16020%2C1589125915820.default.1590422095665) newer than current time (1590425694407 < 1590425696064), probably a clock skew**

**2020-05-26 00:54:54,407 WARN [master,16000,1589125905790\_ChoreService\_1] cleaner.TimeToLiveLogCleaner: Found a log (hdfs://master:9000/hbase/oldWALs/slave2%2C16020%2C1589125917741.default.1590422097831) newer than current time (1590425694407 < 1590425698330), probably a clock skew**

**2020-05-26 01:54:54,576 WARN [master,16000,1589125905790\_ChoreService\_1] cleaner.TimeToLiveLogCleaner: Found a log (hdfs://master:9000/hbase/oldWALs/slave1%2C16020%2C1589125915820.default.1590425696031) newer than current time (1590429294576 < 1590429296507), probably a clock skew**

**2020-05-26 02:54:57,137 WARN [master,16000,1589125905790\_ChoreService\_1] cleaner.TimeToLiveLogCleaner: Found a log (hdfs://master:9000/hbase/oldWALs/slave2%2C16020%2C1589125917741.default.1590429298730) newer than current time (1590432897137 < 1590432899224), probably a clock skew**

### 实验任务四：查看Hive告警信息

Hive的日志文件存储在/tmp/hadoop目录下，切换到该目录，并执行命令ll,显示如下。

[root@master hadoop]# cd /tmp/hadoop

[root@master hadoop]# tail -1000f hive.log |grep INFO

**2020-05-26T11:05:27,968 INFO [org.apache.hadoop.hive.common.JvmPauseMonitor$Monitor@37d3e140]: common.JvmPauseMonitor (JvmPauseMonitor.java:run(194)) - Detected pause in JVM or host machine (eg GC): pause of approximately 4923ms**

**2020-05-26T15:25:31,520 INFO [org.apache.hadoop.hive.common.JvmPauseMonitor$Monitor@37d3e140]: common.JvmPauseMonitor (JvmPauseMonitor.java:run(194)) - Detected pause in JVM or host machine (eg GC): pause of approximately 4439ms**

**2020-05-26T17:17:34,048 INFO [org.apache.hadoop.hive.common.JvmPauseMonitor$Monitor@37d3e140]: common.JvmPauseMonitor (JvmPauseMonitor.java:run(194)) - Detected pause in JVM or host machine (eg GC): pause of approximately 4689ms**

**2020-05-26T19:36:58,155 INFO [org.apache.hadoop.hive.common.JvmPauseMonitor$Monitor@37d3e140]: common.JvmPauseMonitor (JvmPauseMonitor.java:run(194)) - Detected pause in JVM or host machine (eg GC): pause of approximately 4892ms**

Stderr（标准错误），该标准IO流通过预定义文件指针 stderr 加以引用，且该流引用的文件与文件描述符 STDERR\_FILENO 所引用的相同。

[root@master hadoop]# cd /tmp/hadoop

[root@master hadoop]# tail -1000f stderr |grep ERROR

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**

**ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console.**