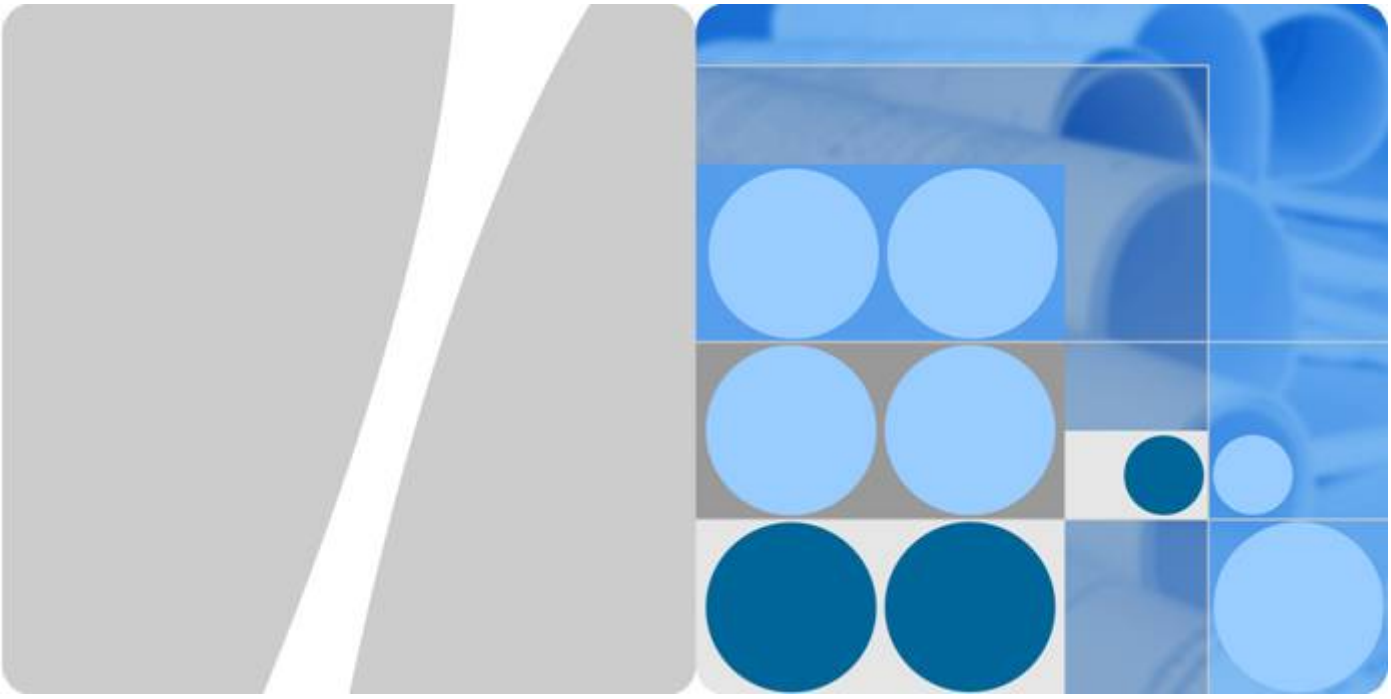


EulerOS V2.0SP3 调试指南（for ARM64）

文档版本　01

发布日期　2019-08-12

华为技术有限公司

商标声明


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

注意

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

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

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： http://www.huawei.com

客户服务邮箱： support@huawei.com

客户服务电话： 4008302118

目 录

1 kbox

1.1 概述

本章主要介绍Kbox（内核黑匣子）基础概念、结构，以及一些软硬件需求和约束限制等。

1.1.1 Kbox 简介

本节介绍Kbox的背景信息和一些基础概念，帮助用户了解Kbox。

概述

linux内核比较复杂，且各个模块之间联系紧密，缺少有效的维护工具，进一步增加了维护难度。虽然内核有klogd和syslogd等日志记录系统，但在一些异常情况下，如系统panic（系统发生严重异常）、oom（内存溢出）、die/oops（内核内存访问异常）、rlock（死锁）等，无法（或者来不及）记录这些信息，进而无法对这些问题的根因进行定位。

针对上述情况，为了挽救丢失的内核日志，EulerOS提供了内核黑匣子（Kernel black box）特性。类似在飞行系统中常用的“黑匣子”，在系统异常触发的时候记录重要信

息，并通过某种特殊渠道（非易失性存储）把这些信息保存下来，供维护人员用来分析发生异常时的系统状态。

特性简介

Kbox提供一种记录内核信息的机制，在系统异常发生时，记录下内核的重要信息，并把这些信息保存到非易失存储介质中。维护人员可通过这些信息，得到发生异常时的具体情况，进而分析发生异常的原因，支撑问题定位。

例如：当panic事件发生时，Kbox将内核产生的异常信息收集并存储在临时存储区（region）。当信息全部收集完成后，Kbox会将临时存储区的信息转储到非易失性设备中（如NVRAM ）或通过kdump保存到指定目录中，供用户查看。

注意

kbox只收集内核打印信息及导致系统异常的函数调用关系，不会保存用户敏感数据。如需关闭kbox功能，请参见**命令参考**。

Kbox 组成

内核黑匣子模块主要分为以下三个部分，详细信息如**表1-1**所示。

表 1-1 黑匣子组成

组成	说明
内核黑匣子	管理非易失性设备，抓取并存储异常事件产生的信息。
非易失性设备驱动	为Kbox提供读写接口。Kbox启动时加载相应的驱动模块，将存储设备注册到Kbox中。
非易失性设备	用于存储异常事件产生的信息。异常事件发生时，Kbox会将异常信息转储到非易失性设备中 。若硬件上不存在非易失性设备，kbox的日志先保存在预留内存中，通过kdump来保存到磁盘中。

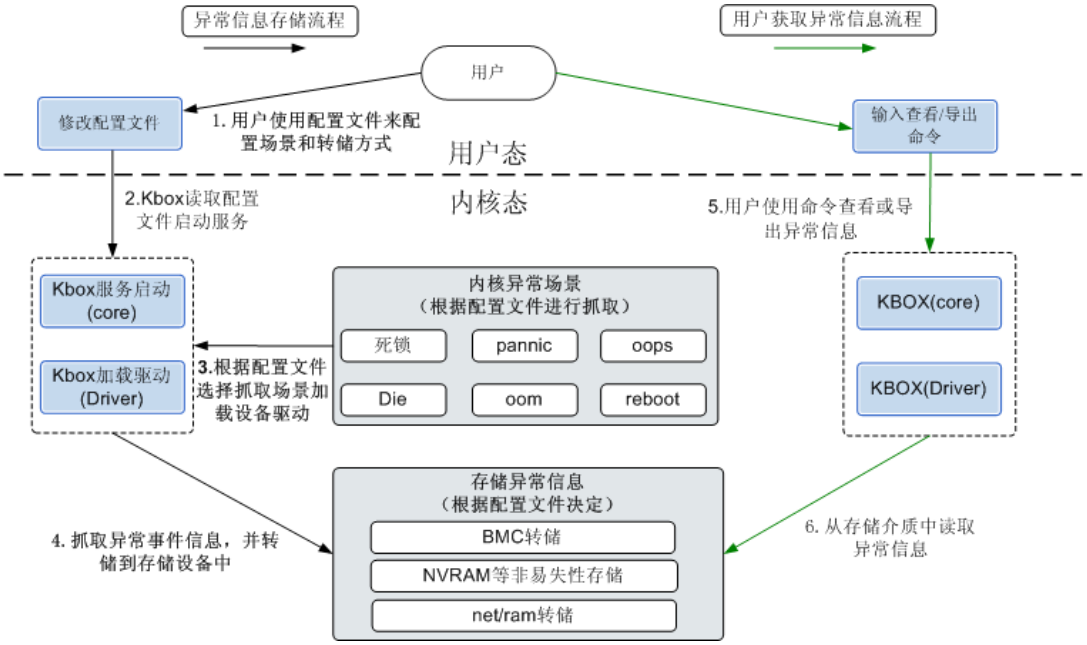
1.1.2 Kbox 结构

本节介绍Kbox的结构以及使用流程。

结构说明

Kbox整体结构和业务流程如下图所示。

图 1-1 Kbox 结构



说明

- 当前存储异常信息暂不支持BMC转储、net/ram转储。
- 当前版本仅支持OOM、OOPS（Die）、Panic、死锁这几种异常场景。

流程说明

1. 修改配置文件。产品根据自己的实际情况，通过配置文件（/etc/kbox/config）配置自己的产品信息、异常抓取场景和转储设备。
2. Kbox以服务的方式启动，读取配置文件（/etc/kbox/config）后，把产品信息和异常场景以模块参数的方式传递给Kbox内核模块。
3. 根据配置文件（/etc/kbox/config）中存储设备参数的配置，Kbox加载相应的存储设备驱动，将存储设备注册到Kbox内核模块。
4. 当某个异常事件发生并被Kbox内核模块抓取后，先记录异常情况相应的信息到对应的临时存储区域（region），然后刷新到存储设备。
5. 具有root权限的用户通过Kbox提供的日志导出命令，将存储设备中的异常信息导出到“/var/log/kbox”目录下。
6. 用户读取异常信息文件，获取异常信息内容，进行问题定位。

1.1.3 软硬件需求

本节介绍Kbox的软硬件需求。

- 软件需求

  仅支持EulerOS操作系统。
- 硬件需求

  Kbox转储功能必须依赖产品提供的转储介质，当前支持NVRAM、hmem、pcie。

  说明

  EulerOS RAS版本在启动参数中添加“node0_reserve_kbox_mem=16M”，预留kbox内存；其他版本使用“reserve_kbox_mem=16M”预留kbox内存。系统启动时，默认已经设置。

1.2 功能说明

本章节介绍了内核黑匣子支持的异常事件场景和抓取的信息，以及对抓取内容的解析。

1.2.1 约束限制

本节介绍Kbox使用时会遇到的一些限制约束。

说明

禁止使用Kbox特性导出敏感信息。

- 当异常事件并发触发时，Kbox只记录最早发生的异常事件，并转储到存储设备。例如：当系统发生panic、oops或者oom等事件后，如果又有同类panic、oops等事件发生，Kbox不做记录，只打印提示信息到系统日志中 。
- Kbox在记录过程中，如果当前CPU上有中断到来，Kbox执行进程被打断，且在中断中又触发异常事件，此时Kbox只打印提示信息到系统日志中，不记录日志信息（包括之前和中断又触发的异常事件）到存储设备。
- 在使用命令行os_kbox_config导出Kbox记录的转储信息时，系统发生panic、oops或者oom等异常事件，由于并发对锁的获取，Kbox可能会概率性出现转储信息不完整的现象。
- 栈向下溢出场景下，Stack获取到信息全为0。栈刚好等于16k时，Stack获取到信息则为空。
- kbox导出日志功能，只能在第一次启动的时候导出，重启kbox服务后，由于kbox记录的标记消失，再次导出日志会提示无日志。
- 虚拟机中kbox使用的普通内存保存日志，系统重启后，可能存在内存重新布局划分，kbox日志丢失。

1.2.2 提供系统 panic 信息

本节主要介绍当系统由于某种异常，直接或间接调用panic函数时，黑匣子所记录的异常信息。

功能概述

产品/平台业务软件或操作系统本身可能因为某种特殊异常，导致操作系统内核发生严重错误。Kbox能够将panic事件发生的时间、调用轨迹等异常信息记录到存储设备中，方便维护人员定位。

信息清单

记录的异常panic信息包含三个部分内容。

1. panic异常信息，最多记录信息容量为32K。该记录区主要内容包括：
   - 内核异常发生的时间(UTC时间)。
   - panic发生的原因。
   - 异常进程PID、异常进程TGID及进程名称。
   - 内核函数调用轨迹及栈信息（栈信息最多打印150行）。

- 当前进程命令行信息。
- 系统环境变量。
- 寄存器信息。

2. 内核消息打印日志，最多记录信息容量为64K。该记录区主要内容包括：
   - 内核异常发生的时间(UTC时间)。
   - 异常发生时最近64K日志信息，即内核打印到循环缓冲区中的最后64K日志信息。
3. 控制台打印日志，最多记录信息容量为32K。该记录区主要内容包括：
   - 内核异常发生的时间(UTC时间)。
   - 其他内核打印日志消息。

   说明

   如果当前其他进程没有向控制台打印日志，收集的控制台打印日志为空。

信息举例

在日志信息文件中，包含如下内容：

1. panic异常信息

```
//panic异常存储区，当前panic日志位于panic存储区索引为0
*****area type:panic - location in panic area:0*****
---------KBOX_START----------
//内核异常发生的时间，该时间为UTC时间，北京时间=UTC时间+8小时
panic time:20180108101725-b8cd7
//panic 原因
panic reason:call panic() function in process context.
//异常进程PID、异常进程TGID及进程名称
process: bash (pid: 6127, tgid: 6127) on CPU: 2
//函数调用轨迹
Call Trace:
[<ffff7ffffc280db8>] kbox_dump_backtrace+0x0/0x110 [kbox]
[<ffff7ffffc281f18>] kbox_show_call_trace+0x188/0x228 [kbox]
[<ffff7ffffc279ef8>] kbox_show_task_kernel_info+0xe8/0x210 [kbox]
[<ffff7ffffc27a084>] kbox_print_specified_tasks+0x64/0x80 [kbox]
[<ffff7ffffc2741d0>] kbox_panic_notifier_callback+0x1d0/0x290 [kbox]
[<ffff8000000f0ed4>] notifier_call_chain+0x5c/0xa0
[<ffff8000000f0f64>] atomic_notifier_call_chain+0x24/0x30
[<ffff8000009ebb68>] panic+0x178/0x2ac
[<ffff7ffffc36fa38>] dev_wr_actions+0xb98/0xf40 [kpgen_kbox]
[<ffff7ffffc36fe9c>] dev_wr_handler+0xbc/0x140 [kpgen_kbox]
[<ffff80000026023c>] __vfs_write+0x4c/0x120
[<ffff800000260c34>] vfs_write+0x9c/0x1a8
[<ffff800000261888>] SyS_write+0x68/0xd8
Stack:
stack grow form: ffff800335937ac0 to ffff800335938000
ffff800334920c40 0000000000000000 ffff7ffffc2889e8 0000808080808080
ffff800335937b00 ffff7ffffc27a084 ffff800334920c40 ffff800000fe7000
ffff800335937b20 ffff7ffffc2741d0 ffff7ffffc283c30 0000000080000000
ffff800335937bd0 ffff8000000f0ed4 ffff80000100edc0 00000000fffffffd
0000000000000000 ffff80000112ca38 0000000000000000 ffff800335937c08
3830313038313032 622d353237313031 ffff800037646338 0000000000000015
ffff800335937bc0 ffff800000098228 00000000000f423c ffff800000fe7b30
ffff800000fe7000 ffff8000001ddc10 ffff80000112c000 ffff800000098208
ffff800335937bd0 cb88537fdc8ba32c ffff800335937c10 ffff8000000f0f64
ffff80000112ca10 ffff80000112c000 ffff7ffffc371000 ffff8000001ddc10
ffff80000112c000 000000000000000b ffff800335937c20 ffff8000009ebb68
ffff800335937cf0 ffff7ffffc36fa38 ffff7ffffc372000 ffff800000fe7000
0000000000000000 ffff8000001ddc10 0000000000000005 0000000000000015
ffff800335937cf0 ffff800335937cf0 ffff800335937cb0 00000000ffffffc8
ffff800335937c90 ffff800335937cf0 ffff800335937cf0 ffff800335937cb0
00000000ffffffc8 cb88537fdc8ba32c 0000ffffac26d000 0000000000000000
cb88537fdc8ba32c 0000000000000000 0000000000000000 0000000000000000
```

```
ffff800000605800 ffff800000fe7b30 ffff800335937d30 ffff7ffffc36fe9c
0000000000000005 ffff800000fe7000 ffff800335937ea0 ffff800335937ea0
...
<pid:6127:6127:bash>
<basic>
bash           R  running 2      0  6127   6127   6124 NA-e NA-y NA-o (NOTLB)
</basic>
//当前进程命令行信息
<cmdline>
-bash
</cmdline>
<env>
USER=root LOGNAME=root HOME=/root PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin
MAIL=/var/mail/root SHELL=/bin/bash SSH_CLIENT=192.168.65.85 59344 22
SSH_CONNECTION=192.168.65.85 59344 192.168.3.233 22 SSH_TTY=/dev/pts/0 TERM=xterm
XDG_SESSION_ID=1 XDG_RUNTIME_DIR=/run/user/0
</env>
<pt_regs>
</pt_regs>
<backtrace>
task bash is 64bit app
kbox_arch_stack_backtrace=ffff800335938000
</backtrace>
<stack_dump:ffff800335937ad0:ffff8003359376d0:400:0>

</stack_dump>
</pid:6127>
```

2. 内核消息打印日志

```
message:
*****area type:message - location in message area:2*****
//内核异常发生的时间，该时间为UTC时间，北京时间=UTC时间+8小时
panic time:20180108101725-b8cd7
[    0.      0] GIC: using LPI property table @0x0000000379cc0000
[    0.      0] ITS: Allocated 1792 chunks for LPIs
[    0.      0] GICv3: CPU0: found redistributor 0 region 0:0x00000000080a0000
[    0.      0] CPU0: using LPI pending table @0x0000000379cd0000
[    0.      0] Architected cp15 timer(s) running at 50.00MHz (virt).
[    0.      0] clocksource arch_sys_counter: mask: 0xffffffffffffff max_cycles: 0xb8812736b,
max_idle_ns: 440795202655 ns
[    0.      1] sched_clock: 56 bits at 50MHz, resolution 20ns, wraps every 4398046511100ns
[    0.   1566] Console: colour dummy device 80x25
[    0.   2397] console [tty0] enabled
[    0.   3035] bootconsole [pl11] disabled
[    0.   3749] kmemleak: Kernel memory leak detector disabled
[    0.   3764] Calibrating delay loop (skipped), value calculated using timer frequency..
100.00 BogoMIPS (lpj=200000)
[    0.   3770] pid_max: default: 32768 minimum: 301
[    0.   3793] Security Framework initialized
[    0. 19135] Dentry cache hash table entries: 2097152 (order: 12, 16777216 bytes)
[    0.127604] Inode-cache hash table entries: 1048576 (order: 11, 8388608 bytes)
[    0.168332] Mount-cache hash table entries: 32768 (order: 6, 262144 bytes)
[    0.168369] Mountpoint-cache hash table entries: 32768 (order: 6, 262144 bytes)
[    0.181797] Initializing cgroup subsys blkio
[    0.181803] Initializing cgroup subsys memory
[    0.181813] Initializing cgroup subsys devices
[    0.181818] Initializing cgroup subsys freezer
[    0.181822] Initializing cgroup subsys net_cls
[    0.181826] Initializing cgroup subsys perf_event
[    0.181831] Initializing cgroup subsys net_prio
[    0.181836] Initializing cgroup subsys hugetlb
[    0.181847] ftrace: allocating 33040 entries in 130 pages
[    0.276924] hw perfevents: enabled with arm/armv8-pmuv3 PMU driver, 7 counters available
......
[    9.968716] ip_set: protocol 6
[    9.981637] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[    9.981941] 8021q: adding VLAN 0 to HW filter on device eth0
[  707.978508] kpgenm: call panic() function in process context.
[  707.988306] Kernel panic - not syncing: call panic() function in process context.
```



```
[  707.989694] CPU: 2 PID: 6127 Comm: bash Tainted: G           OE
4.1.44-03.35.vhulk1711.1.1.aarch64 #1
[  707.991165] Hardware name: QEMU KVM Virtual Machine, BIOS 0.0.0 02/06/2015
[  707.992235] Call trace:
[  707.992634] [<ffff80000008ae10>] dump_backtrace+0x0/0x1a0
[  707.993476] [<ffff80000008afd0>] show_stack+0x20/0x28
[  707.994271] [<ffff8000009eef1c>] dump_stack+0x98/0xb8
[  707.995068] [<ffff8000009ebb3c>] panic+0x14c/0x2ac
[  707.995822] [<ffff7ffffc36fa38>] dev_wr_actions+0xb98/0xf40 [kpgen_kbox]
[  707.996862] [<ffff7ffffc36fe9c>] dev_wr_handler+0xbc/0x140 [kpgen_kbox]
[  707.997892] [<ffff80000026023c>] __vfs_write+0x4c/0x120
[  707.998710] [<ffff800000260c34>] vfs_write+0x9c/0x1a8
[  707.999510] [<ffff800000261888>] SyS_write+0x68/0xd8
[  708.    285] SMP: stopping secondary CPUs
[  708.    920] kbox: catch panic event.
```

3. 控制台打印日志

```
console:
###num:0, record len:254
//内核异常发生的时间，该时间为UTC时间，北京时间=UTC时间+8小时
panic time:20180108101725-b8cd7
[  708.001834] collected_len = 28142, LOG_BUF_LEN_LOCAL = 32768
[  708.110716] [kbox drv] flush data done. addr: 0xffff800358800000, size: 0x1000000
panic time:20180108101725-b8cd7
```

1.2.3 提供系统 oom 信息

本节主要介绍当内存不足发生时内核黑匣子所记录的异常信息。

功能概述

产品/平台业务软件或操作系统本身可能因为某种特殊原因导致内存空余不足，触发oom事件。Kbox能够将oom事件发生的时间、发生oom进程信息、系统进程信息等异常信息记录到存储设备中，方便维护人员定位。

信息清单

记录的异常oom信息包含三个部分内容。

1. oom异常信息，最多记录信息容量为256K。该记录区主要内容包括：
   - 内核异常发生的时间(UTC时间)。
   - 当前进程的pid、进程名称、内核版本、cpu号、cmd。

     说明

     cmd信息记录的是进程的cmdline参数，这里kbox只打印前255个字符。
   - 异常进程的调用轨迹及栈信息（栈信息最多打印150行）。
   - 内存统计信息。
   - slab分配信息。
   - 当前系统中进程信息。
   - vmalloc信息。
   - rootfs/ramfs/tmpfs等内存文件系统中的文件信息。
   - rootfs/ramfs/tmpfs等内存文件系统中的文件占用内存的信息。
   - oom后的操作，0：不做任何操作 1：调用panic。
2. 内核消息打印日志，最多记录信息容量为64K。该记录区主要内容包括：
   - 内核异常发生的时间(UTC时间)。

- 异常发生时最近64K日志信息，即内核打印到循环缓冲区中的最后64K日志信息。

3. 控制台打印日志，最多记录信息容量为32K。该记录区主要内容包括：
   - 内核异常发生的时间(UTC时间)。
   - 其他内核打印日志消息。

信息举例

在日志信息文件中，包含如下内容：

1. oom异常信息

```
// oom异常存储区，当前oom日志记录1个
area type: 0-panic 1-reboot 2-emerge 3-intermit 4-oom 5-oops/die 6-rlock 7-message 8-console
---------------area type:4, record num:1, unit_size:64 ------------


###num:0, record len:83407
---------KBOX_START----------
// 内核异常发生的时间，该时间为UTC时间，北京时间=UTC时间+8小时
oom time:20160218155821-e0387
//当前触发oom进程的pid，进程名及内核版本和cpu号
Current Pid: 1621, comm: kworker/2:3 Tainted: G           OE
4.1.44-03.35.vhulk1711.1.1.aarch64 #1
//异常进程函数调用轨迹
Call Trace:
 [<ffffffffa037ca53>] kbox_dump_oom+0x13/0x30 [kbox]
 [<ffffffffa037bb17>] kbox_oom_callback+0x107/0x210 [kbox]
 [<ffffffff810a8c20>] ? wake_up_state+0x20/0x20
 [<ffffffff8109ffe8>] ? __wake_up_common+0x58/0x90
 [<ffffffff8160a2fc>] notifier_call_chain+0x4c/0x70
 [<ffffffff8109d07d>] __blocking_notifier_call_chain+0x4d/0x70
 [<ffffffff8109d0b6>] blocking_notifier_call_chain+0x16/0x20
 [<ffffffff81159e6e>] out_of_memory+0x4e/0x4f0
 [<ffffffff81390b1c>] ? tty_ldisc_deref+0x3c/0xa0
 [<ffffffff81394fad>] moom_callback+0x4d/0x50
 [<ffffffff8108f445>] process_one_work+0x175/0x430
 [<ffffffff8109028b>] worker_thread+0x11b/0x3a0
 [<ffffffff81090170>] ? manage_workers.isra.25+0x2a0/0x2a0
 [<ffffffff81096fbf>] kthread+0xcf/0xe0
 [<ffffffff81096ef0>] ? insert_kthread_work+0x40/0x40
 [<ffffffff8160e858>] ret_from_fork+0x58/0x90
 [<ffffffff81096ef0>] ? insert_kthread_work+0x40/0x40
oom stack:
 ffff8801385bfbf0 ffff8801385bfc48 ffff8801385bfc6c 0000000000000000
 ffff8801385bfdc8 0000000000000000 ffff8801385bfc48 ffffffffa037c04a
 0000000000000002 0000000045746160 0000000000000000 ffff8801385bfc58
 ffffffffa037ca53 ffff8801385bfce8 ffffffffa037bb17 36313032810a8c32
 3835353138313230 37383330652d3132 0000000000000000 ffff880035ec2280
 ffffffff810a8c20 0000000000000000 0000000000000000 0000000045746160
 ffff8801385bfcf8 ffffffff8109ffe8 0000000100000000 0000000000000000
 0000000045746160 00000000ffffffff 0000000000000000 ffff8801385bfd20
 ffffffff8160a2fc ffffffff81977f80 0000000000000000 0000000000000000
 ffff8801385bfdc8 00000000ffffffff ffff8801385bfd60 ffffffff8109d07d
 0000000000000000 0000000000000000 0000000000000001 ffff88013ec92f00
 ffff88013efd8300 00000000000000d0 ffff8801385bfd70 ffffffff8109d0b6
 ffff8801385bfe08 ffffffff81159e6e 00000000fffc4c8b 0000000000000000
 ffff8801385bfde0 0000000000000286 0000000000000286 0000000000000286
 ffff8801385bfdd0 ffffffff81390b1c 0000000000000282 0000000000000000
 ffff8801385bfe18 0000000045746160 ffffffff819bd9a0 ffff88013846a000
 ffff88013ec92f00 ffff88013ec96f00 0000000000000040 ffff8801385bfe18
 ffffffff81394fad ffff8801385bfe60 ffffffff8108f445 000000003ec92f00
 0000000000000000 ffff88013ec92f00 ffff88013ec92f18 ffff88013846a030
 ffff880138589700 ffff88013846a000 ffff8801385bfec0 ffffffff8109028b
 ffff8801385bffd8 ffff8801385bffd8 ffff880138589700 ffff880138589700
 ffff880138589700 ffff880138bafd38 ffff88013846a000 ffffffff81090170
 0000000000000000 0000000000000000 ffff8801385bff48 ffffffff81096fbf
```

```
 0000000000000000 0000000000000000 ffff88013846a000 0000000000000000
 0000000000000000 ffff8801385bfef8 ffff8801385bfef8 ffff880100000000
 ffff880100000000 ffff8801385bff18 ffff8801385bff18 0000000045746160
 ffffffff81096ef0 0000000000000000 0000000000000000 ffff880138bafd38
 ffffffff8160e858 0000000000000000 0000000000000000 0000000000000000
 0000000000000000 ffff880138bafd38 ffffffff81096ef0 0000000000000000
 0000000000000000 0000000000000000 0000000000000000 0000000000000000
 0000000000000000 0000000000000000 0000000000000000 0000000000000000
 ffffffffffffffff 0000000000000000 0000000000000010 0000000000000202
 ffff8801385bff58 0000000000000018
//内存统计信息
*******  meminfo  *******
MemTotal:     3768752 kB
MemFree:      3522112 kB
TotalHigh:          0 kB
FreeHigh:           0 kB
Buffers:          764 kB
SwapTotal:     4063228 kB
SwapFree:      4063228 kB
//slab分配信息
*******  slabinfo  *******
# name            <active_objs> <num_objs> <objsize> <objperslab> <pagesperslab>: tunables
<limit> <batchcount> <sharedfactor>: slabdata <active_slabs> <num_slabs> <sharedavail>
nf_conntrack_ffffffff819dcb40     50     50    320   25    2 : tunables    0    0    0 :
slabdata      2      2      0
xfs_icr                0      0    144   28    1 : tunables    0    0    0 :
slabdata      0      0      0
xfs_ili                104    104    152   26    1 : tunables    0    0    0 :
slabdata      4      4      0
xfs_inode              1760   1760   1024   16    4 : tunables    0    0    0 :
slabdata    110    110      0
xfs_efd_item             80     80    400   20    2 : tunables    0    0    0 :
slabdata      4      4      0
xfs_da_state             64     64    488   16    2 : tunables    0    0    0 :
slabdata      4      4      0
xfs_btree_cur            76     76    208   19    1 : tunables    0    0    0 :
slabdata      4      4      0
xfs_log_ticket           88     88    184   22    1 : tunables    0    0    0 :
slabdata      4      4      0
scsi_cmd_cache           72     72    448   18    2 : tunables    0    0    0 :
slabdata      4      4      0
kcopyd_job               0      0   3312    9    8 : tunables    0    0    0 :
slabdata      0      0      0
dm_uevent                0      0   2608   12    8 : tunables    0    0    0 :
slabdata      0      0      0
dm_rq_target_io          0      0    424   19    2 : tunables    0    0    0 :
slabdata      0      0      0
UDPLITEv6                0      0   1152   28    8 : tunables    0    0    0 :
slabdata      0      0      0
UDPv6                    84     84   1152   28    8 : tunables    0    0    0 :
slabdata      3      3      0
tw_sock_TCPv6            0      0    256   16    1 : tunables    0    0    0 :
slabdata      0      0      0
TCPv6                    16     16   2048   16    8 : tunables    0    0    0 :
slabdata      1      1      0
cfq_queue                102    102    232   17    1 : tunables    0    0    0 :
slabdata      6      6      0
bsg_cmd                  0      0    312   26    2 : tunables    0    0    0 :
slabdata      0      0      0
mqueue_inode_cache       18     18    896   18    4 : tunables    0    0    0 :
slabdata      1      1      0
hugetlbfs_inode_cache    52     52    608   26    4 : tunables    0    0    0 :
slabdata      2      2      0
configfs_dir_cache       0      0     88   46    1 : tunables    0    0    0 :
slabdata      0      0      0
dquot                    0      0    256   16    1 : tunables    0    0    0 :
slabdata      0      0      0
kioctx                   0      0    576   28    4 : tunables    0    0    0 :
slabdata      0      0      0
```



```
pid_namespace                  0      0   2176   15    8 : tunables    0    0    0 :
slabdata      0      0      0
posix_timers_cache             0      0    248   16    1 : tunables    0    0    0 :
slabdata      0      0      0
UDP-Lite                       0      0   1024   16    4 : tunables    0    0    0 :
slabdata      0      0      0
......

//当时系统的进程信息
//说明：cmd信息记录的是进程的cmdline参数，这里kbox只打印前255个字符
*******  kbox show all tasks  *******
[ pid ]   uid  ppid  tgid total_vm      rss(pages) cpu stat name cmd[    1]     0     0
1    10275             889   1    S systemd /usr/lib/systemd/systemd --switched-root --
system --deserialize 20[  243]     0     1   243     9204            2186   3    S systemd-
journal /usr/lib/systemd/systemd-journald[  266]     0     1   266    10788
524   2    S systemd-udevd /usr/lib/systemd/systemd-udevd
[  349]     0     1   349    39155            2398   1    S rsyslogd /usr/sbin/rsyslogd -
n[  363]    81     1   363     6617             365   0    S dbus-daemon /bin/dbus-daemon --
system --address=systemd: --nofork --nopidfile --systemd-activation[  398]     0     1
398     6613             440   3    S systemd-logind /usr/lib/systemd/systemd-logind
[  408]     0     1   408    27509             208   2    S agetty /sbin/agetty --keep-baud
115200 38400 9600 ttyS0 vt220
[  414]     0     1   414    31583             393   3    S crond /usr/sbin/crond -n
[  421]     0     1   421    24241             673   1    S login login -- root
[  680]     0     1   680    17941             539   3    S sendmail sendmail: accepting
connect
[  845]    51     1   845    16801             463   3    S sendmail sendmail: Queue
runner@01:00:00 for /var/sp
[  880]     0     1   880    16631             314   0    S sshd /usr/sbin/sshd
[ 2541]     0     1  2541    24144            3261   2    S dhclient /sbin/dhclient -H
localhost -1 -q -lf /var/lib/dhclient/dhclient--eth0.lease -pf /var/run/dhclient-eth0.pid
eth0
[ 2841]     0   421  2841    28875             541   0    R bash -bash

total real task number: 14
*****************Start oom extend info.*****************
Vmallocinfo Start >>>>>>>>>>>>>>>>>>>>//通过内核接口申请的内存
0xffffc90000000000-0xffffc90000002000       8192 hpet_enable+0x2d/0x2e9 phys=0xfed00000
ioremap
0xffffc90000002000-0xffffc90000403000    4198400 alloc_large_system_hash+0x171/0x239
pages=1024 vmalloc vpages
0xffffc90000403000-0xffffc90000406000      12288 alloc_large_system_hash+0x171/0x239 pages=2
vmalloc
0xffffc90000406000-0xffffc90000607000    2101248 alloc_large_system_hash+0x171/0x239
pages=512 vmalloc
0xffffc90000608000-0xffffc9000060b000      12288 acpi_os_map_memory+0xea/0x142
phys=0xbffdf000 ioremap
0xffffc9000060c000-0xffffc9000060f000      12288 acpi_os_map_memory+0xea/0x142
phys=0xbffe0000 ioremap
0xffffc90000610000-0xffffc90000612000       8192 acpi_os_map_memory+0xea/0x142
phys=0xfed00000 ioremap
0xffffc90000612000-0xffffc90000653000     266240 alloc_large_system_hash+0x171/0x239
pages=64 vmalloc
0xffffc90000653000-0xffffc900006d4000     528384 alloc_large_system_hash+0x171/0x239
pages=128 vmalloc
0xffffc900006d4000-0xffffc900006e5000      69632 alloc_large_system_hash+0x171/0x239
pages=16 vmalloc
0xffffc900006e5000-0xffffc900006f6000      69632 alloc_large_system_hash+0x171/0x239
pages=16 vmalloc
0xffffc900006f6000-0xffffc90000707000      69632 alloc_large_system_hash+0x171/0x239
pages=16 vmalloc
0xffffc90000708000-0xffffc9000070a000       8192 cirrus_device_init+0x86/0x130 [cirrus]
phys=0xfebf4000 ioremap
0xffffc9000070a000-0xffffc9000070c000       8192 dm_table_create+0x9e/0x130 [dm_mod] pages=1
vmalloc
0xffffc9000070c000-0xffffc9000072d000     135168 raw_init+0x41/0x141 pages=32 vmalloc
0xffffc9000072d000-0xffffc90000736000      36864 drm_ht_create+0x55/0x80 [drm] pages=8
vmalloc
0xffffc90000736000-0xffffc90000738000       8192 dm_table_create+0x9e/0x130 [dm_mod] pages=1
```



```
vmalloc
0xffffc90000739000-0xffffc9000073c000      12288 cirrusfb_create+0xde/0x390 [cirrus] pages=2
vmalloc
0xffffc9000073c000-0xffffc9000073e000       8192 swap_cgroup_swapon+0x48/0x160 pages=1
vmalloc
0xffffc9000073e000-0xffffc90000740000       8192 ebt_register_table+0xa2/0x380 [ebtables]
pages=1 vmalloc
0xffffc90000740000-0xffffc90000761000     135168 pci_ioremap_bar+0x46/0x80 phys=0xfebc0000
ioremap
0xffffc90000761000-0xffffc900009a2000    2363392 cirrusfb_create+0xde/0x390 [cirrus]
pages=576 vmalloc vpages
......
Vmallocinfo End <<<<<<<<<<<<<<<<<<<<

//内存文件系统
Filesystem            1K-blocks    Used   Available Use(%)   Mounted on
tmpfs                  1884376        0  1884376        0%   /dev/shm
tmpfs                  1884376     8504  1875872        0%   /run
tmpfs                  1884376        0  1884376        0%   /sys/fs/cgroup
********    mem info     ********?Total:                3768752 kB
Total free:           3521908 kB
User space:            157772 kB
Mlock:                      0 kB
Kernel space:           89072 kB
Bootmem reserved:      409164 kB
HugePages_Total:            0 kB
HugePages_Free:             0 kB
HugePages_Rsvd:             0 kBHugePages_Surp:             0 kB
Hugepagesize:            2048 kB
//占用内存的文件，从大到小
******     pagecache_info:     ******
  /run/log/journal/ad0c8c7dc64c4ecb860400450d3d0825/system.journal : nrpages = 1971.
  /run/udev/data/c116:33 : nrpages = 1.
  /run/udev/data/c116:3 : nrpages = 1.
  /run/udev/data/b8:2 : nrpages = 1.  /run/systemd/system/session-1.scope.d/90-
SendSIGHUP.conf : nrpages = 1.  /run/udev/data/b253:1 : nrpages = 1.
  /run/mcelog.pid : nrpages = 1.
  /run/udev/data/n2 : nrpages = 1.  /run/systemd/system/session-1.scope.d/90-Slice.conf :
nrpages = 1.  /run/udev/data/b8:1 : nrpages = 1.
  /run/syslogd.pid : nrpages = 1.
  /run/udev/data/b8:0 : nrpages = 1.
  /unknown : nrpages = 1.
  /run/udev/data/b11:0 : nrpages = 1.
  /run/crond.pid : nrpages = 1.
  /run/udev/data/c13:33 : nrpages = 1.  /run/systemd/system/session-1.scope : nrpages =
1.  /run/udev/data/c13:67 : nrpages = 1.
  /run/auditd.pid : nrpages = 1.
  ......
*****************End oom extend info.*****************
//oom异常记录后的操作(0:返回调用链，1:调用panic)
action after oom is:1
---------KBOX_END------------
```

2. 内核消息打印日志

```
---------------area type:7, record num:1, unit_size:64 ------------


###num:0, record len:39434
oom time:20160218155821-e0387
[    0.182489] Initializing cgroup subsys net_cls
[    0.182493] Initializing cgroup subsys perf_event
[    0.182498] Initializing cgroup subsys net_prio
[    0.182504] Initializing cgroup subsys hugetlb
[    0.182514] ftrace: allocating 33040 entries in 130 pages
[    0.279601] hw perfevents: enabled with arm/armv8-pmuv3 PMU driver, 7 counters available
[    0.279618] Remapping and enabling EFI services.
[    0.279624]   EFI remap 0x0000000004000000 => 0000000040000000
[    0.279640]   EFI remap 0x0000000009010000 => 0000000044000000
[    0.279649]   EFI remap 0x00000003956f0000 => 0000000044010000
[    0.279690]   EFI remap 0x00000003957b0000 => 00000000440d0000
```

```
[    0.279710]   EFI remap 0x0000000395820000 => 0000000044120000
[    0.279737]   EFI remap 0x0000000395890000 => 0000000044190000
[    0.279757]   EFI remap 0x00000003958e0000 => 00000000441e0000
[    0.279780]   EFI remap 0x0000000395930000 => 0000000044230000
[    0.279800]   EFI remap 0x0000000398c60000 => 0000000044280000
[    0.279833]   EFI remap 0x0000000398d00000 => 0000000044310000
[    0.280674] CPU1: Booted secondary processor
[    0.280677] Detected PIPT I-cache on CPU1
[    0.280690] GICv3: CPU1: found redistributor 1 region 0:0x00000000080c0000
[    0.280748] CPU1: using LPI pending table @0x0000000378370000
[    0.280971] CPU2: Booted secondary processor
[    0.280974] Detected PIPT I-cache on CPU2
[    0.280986] GICv3: CPU2: found redistributor 2 region 0:0x00000000080e0000
[    0.281043] CPU2: using LPI pending table @0x00000003783a0000
[    0.338976] CPU3: Booted secondary processor
[    0.338981] Detected PIPT I-cache on CPU3
[    0.338999] GICv3: CPU3: found redistributor 3 region 0:0x0000000008100000
[    0.339047] CPU3: using LPI pending table @0x00000003783d0000
[    0.339169] Brought up 4 CPUs
[    0.339222] SMP: Total of 4 processors activated.
[    0.339227] CPU: All CPU(s) started at EL1
[    0.339260] alternatives: patching kernel code
[    0.352848] devtmpfs: initialized
[    0.355692] SMBIOS 3.0.0 present.
[    0.355795] clocksource jiffies: mask: 0xffffffff max_cycles: 0xffffffff, max_idle_ns:
7645041785100000 ns
[    0.355822] futex hash table entries: 1024 (order: 4, 65536 bytes)
[    0.356029] xor: measuring software checksum speed
[    0.395963]    8regs     :  4208.000 MB/sec
[    0.436026]    8regs_prefetch:  3977.000 MB/sec
[    0.476091]    32regs    :  4093.000 MB/sec
[    0.516153]    32regs_prefetch:  3864.000 MB/sec
[    0.516159] xor: using function: 8regs (4208.000 MB/sec)
[    0.516211] pinctrl core: initialized pinctrl subsystem
[    0.516550] NET: Registered protocol family 16
[    0.528219] cpuidle: using governor ladder
[    0.540276] cpuidle: using governor menu
[    0.540364] vdso: 2 pages (1 code @ ffff800000fe5000, 1 data @ ffff800000fe4000)
[    0.540374] vdso: 2 pages (1 code @ ffff800000a07000, 1 data @ ffff800000fe4000)
[    0.540511] hw-breakpoint: found 6 breakpoint and 4 watchpoint registers.
[    0.553800] DMA: preallocated 256 KiB pool for atomic allocations
[    0.553870] Serial: AMBA PL011 UART driver
[    0.589201] 9000000.pl011: ttyAMA0 at MMIO 0x9000000 (irq = 38, base_baud = 0) is a PL011
rev1
[    0.707387] console [ttyAMA0] enabled
[    0.863259] raid6: int64x1  gen()  1639 MB/s
[    0.931565] raid6: int64x1  xor()  1354 MB/s
[    0.999923] raid6: int64x2  gen()  2588 MB/s
[    1. 68260] raid6: int64x2  xor()  2011 MB/s
[    1.136581] raid6: int64x4  gen()  3203 MB/s
[    1.204899] raid6: int64x4  xor()  2099 MB/s
[    1.273252] raid6: int64x8  gen()  3581 MB/s
[    1.341555] raid6: int64x8  xor()  2045 MB/s
[    1.409895] raid6: neonx1   gen()  3174 MB/s
[    1.478189] raid6: neonx2   gen()  4896 MB/s
[    1.546504] raid6: neonx4   gen()  6581 MB/s
[    1.614813] raid6: neonx8   gen()  5936 MB/s
[    1.615466] raid6: using algorithm neonx4 gen() 6581 MB/s
[    1.616261] raid6: using intx1 recovery algorithm
[    1.617130] pcie_bus_config value 0 -> 0
[    1.617965] vgaarb: loaded
[    1.618486] SCSI subsystem initialized
[    1.619155] libata version 3.00 loaded.
[    1.619286] usbcore: registered new interface driver usbfs
[    1.620117] usbcore: registered new interface driver hub
[    1.621003] usbcore: registered new device driver usb
[    1.621863] pps_core: LinuxPPS API ver. 1 registered
[    1.622591] pps_core: Software ver. 5.3.6 - Copyright 2005-2007 Rodolfo Giometti
<giometti@linux.it>
```



```
[    1.623927] PTP clock support registered
[    1.624624] EDAC MC: Ver: 3.0.0
[    1.625516] NetLabel: Initializing
[    1.626034] NetLabel:  domain hash size = 128
[    1.626671] NetLabel:  protocols = UNLABELED CIPSOv4
[    1.627448] NetLabel:  unlabeled traffic allowed by default
[    1.628392] Switched to clocksource arch_sys_counter
[    1.680535] FS-Cache: Loaded
[    1.684424] NET: Registered protocol family 2
[    1.685392] TCP established hash table entries: 131072 (order: 8, 1048576 bytes)
[    1.686802] TCP bind hash table entries: 65536 (order: 8, 1048576 bytes)
[    1.687964] TCP: Hash tables configured (established 131072 bind 65536)
[    1.689028] UDP hash table entries: 8192 (order: 6, 262144 bytes)
[    1.690012] UDP-Lite hash table entries: 8192 (order: 6, 262144 bytes)
[    1.691059] NET: Registered protocol family 1
[    1.691857] RPC: Registered named UNIX socket transport module.
[    1.692745] RPC: Registered udp transport module.
[    1.693446] RPC: Registered tcp transport module.
[    1.694133] RPC: Registered tcp NFSv4.1 backchannel transport module.
[    1.695064] PCI: CLS 0 bytes, default 64
[    1.695112] Unpacking initramfs...
[    2.293805] Freeing initrd memory: 17708K
[    2.294690] kvm [1]: HYP mode not available
[    2.295644] Initialise system trusted keyring
[    2.296323] audit: initializing netlink subsys (disabled)
[    2.297204] audit: type=2000 audit(1.960:1): initialized
[    2.298285] HugeTLB registered 1 GB page size, pre-allocated 0 pages
[    2.299214] HugeTLB registered 2 MB page size, pre-allocated 0 pages
[    2.302679] zpool: loaded
[    2.303084] zbud: loaded
[    2.303751] VFS: Disk quotas dquot_6.6.0
[    2.304413] VFS: Dquot-cache hash table entries: 512 (order 0, 4096 bytes)
[    2.306421] NFS: Registering the id_resolver key type
[    2.307171] Key type id_resolver registered
[    2.307790] Key type id_legacy registered
[    2.308385] nfs4filelayout_init: NFSv4 File Layout Driver Registering...
[    2.309430] Installing knfsd (copyright (C) 1996 okir@monad.swb.de).
[    2.310794] Key type cifs.spnego registered
[    2.311421] Key type cifs.idmap registered
[    2.312190] Key type big_key registered
[    2.314072] alg: No test for stdrng (krng)
[    2.314742] NET: Registered protocol family 38
[    2.315410] async_tx: api initialized (async)
[    2.316060] Key type asymmetric registered
[    2.316688] Asymmetric key parser 'x509' registered
[    2.317437] bounce: pool size: 64 pages
[    2.331278] Block layer SCSI generic (bsg) driver version 0.4 loaded (major 250)
[    2.332456] io scheduler noop registered
[    2.333039] io scheduler deadline registered (default)
[    2.333855] io scheduler cfq registered
[    2.334873] pl061_gpio 9030000.pl061: PL061 GPIO chip @0x0000000009030000 registered
[    2.336131] pci_hotplug: PCI Hot Plug PCI Core version: 0.5
[    2.336999] pciehp: PCI Express Hot Plug Controller Driver version: 0.4
[    2.338021] PCI host bridge /pcie@eff0000 ranges:
[    2.338717]    IO 0x3dfe0000..0x3dffffff -> 0x00000000
[    2.339472]   MEM 0x0eff0000..0x3dfdffff -> 0x0eff0000
[    2.340228]   MEM 0x8000000000..0xfffffffffff -> 0x8000000000
[    2.342858] pci-host-generic 3e000000.pcie: PCI host bridge to bus 0000:00
[    2.343866] pci_bus 0000:00: root bus resource [bus 00-1f]
[    2.344689] pci_bus 0000:00: root bus resource [io  0x0000-0x1ffff]
[    2.345614] pci_bus 0000:00: root bus resource [mem 0x0eff0000-0x3dfdffff]
[    2.346613] pci_bus 0000:00: root bus resource [mem 0x8000000000-0xfffffffffff]
[    2.347692] pci 0000:00:00.0: [1b36:0008] type 00 class 0x060000
[    2.347857] pci 0000:00:00.0: of_irq_parse_pci() failed with rc=-19
[    2.348877] pci 0000:00:01.0: [8086:244e] type 01 class 0x060401
[    2.360695] pci 0000:00:01.0: of_irq_parse_pci() failed with rc=-19
[    2.366710] pci_bus 0000:01: busn_res: can not insert [bus 01-ff] under [bus 00-1f]
(conflicts with (null) [bus 00-1f])
[    2.366758] pci 0000:01:00.0: [1b36:0001] type 01 class 0x060400
```



```
[    2.373523] pci 0000:01:00.0: reg 0x10: [mem 0x0f100000-0x0f1000ff 64bit]
[    2.406987] pci 0000:02:02.0: [1af4:1000] type 00 class 0x020000
[    2.409872] pci 0000:02:02.0: reg 0x10: [io  0x0040-0x007f]
[    2.413874] pci 0000:02:02.0: reg 0x14: [mem 0x0f002000-0x0f002fff]
[    2.422175] pci 0000:02:02.0: reg 0x20: [mem 0x800000c000-0x800000ffff 64bit pref]
[    2.426281] pci 0000:02:02.0: reg 0x30: [mem 0xfffc0000-0xffffffff pref]
[    2.426638] pci 0000:02:03.0: [1af4:1004] type 00 class 0x010000
[    2.429300] pci 0000:02:03.0: reg 0x10: [io  0x0000-0x003f]
[    2.431868] pci 0000:02:03.0: reg 0x14: [mem 0x0f001000-0x0f001fff]
[    2.441252] pci 0000:02:03.0: reg 0x20: [mem 0x8000008000-0x800000bfff 64bit pref]
[    2.444155] pci 0000:02:04.0: [1af4:1050] type 00 class 0x038000
[    2.455450] pci 0000:02:04.0: reg 0x14: [mem 0x0f000000-0x0f000fff]
[    2.474339] pci 0000:02:04.0: reg 0x20: [mem 0x8000004000-0x8000007fff 64bit pref]
[    2.481567] pci 0000:02:05.0: [1af4:1002] type 00 class 0x00ff00
[    2.484169] pci 0000:02:05.0: reg 0x10: [io  0x0080-0x009f]
[    2.496483] pci 0000:02:05.0: reg 0x20: [mem 0x8000000000-0x8000003fff 64bit pref]
[    2.499431] pci_bus 0000:02: busn_res: [bus 02-ff] end is updated to 02
[    2.501024] pci_bus 0000:01: busn_res: [bus 01-ff] end is updated to 02
[    2.514993] pci 0000:00:01.0: BAR 14: assigned [mem 0x0f000000-0x0f1fffff]
[    2.517548] pci 0000:00:01.0: BAR 15: assigned [mem 0x8000000000-0x80000fffff 64bit pref]
[    2.518736] pci 0000:00:01.0: BAR 13: assigned [io  0x1000-0x1fff]
[    2.519643] pci 0000:01:00.0: BAR 14: assigned [mem 0x0f000000-0x0f0fffff]
[    2.520671] pci 0000:01:00.0: BAR 15: assigned [mem 0x8000000000-0x80000fffff 64bit pref]
[    2.521870] pci 0000:01:00.0: BAR 13: assigned [io  0x1000-0x1fff]
[    2.522769] pci 0000:01:00.0: BAR 0: assigned [mem 0x0f100000-0x0f1000ff 64bit]
[    2.528483] pci 0000:02:02.0: BAR 6: assigned [mem 0x0f000000-0x0f03ffff pref]
[    2.530343] pci 0000:02:02.0: BAR 4: assigned [mem 0x8000000000-0x8000003fff 64bit pref]
[    2.531551] pci 0000:02:03.0: BAR 4: assigned [mem 0x8000004000-0x8000007fff 64bit pref]
[    2.532786] pci 0000:02:04.0: BAR 4: assigned [mem 0x8000008000-0x800000bfff 64bit pref]
[    2.538775] pci 0000:02:05.0: BAR 4: assigned [mem 0x800000c000-0x800000ffff 64bit pref]
[    2.540718] pci 0000:02:01.0: BAR 0: assigned [mem 0x0f040000-0x0f040fff]
[    2.544110] pci 0000:02:02.0: BAR 1: assigned [mem 0x0f041000-0x0f041fff]
[    2.545172] pci 0000:02:03.0: BAR 1: assigned [mem 0x0f042000-0x0f042fff]
[    2.546172] pci 0000:02:04.0: BAR 1: assigned [mem 0x0f043000-0x0f043fff]
[    2.549608] pci 0000:02:02.0: BAR 0: assigned [io  0x1000-0x103f]
[    2.553674] pci 0000:02:03.0: BAR 0: assigned [io  0x1040-0x107f]
[    2.557009] pci 0000:02:05.0: BAR 0: assigned [io  0x1080-0x109f]
[    2.560254] pci 0000:01:00.0: PCI bridge to [bus 02]
[    2.562128] pci 0000:01:00.0:   bridge window [io  0x1000-0x1fff]
[    2.566713] pci 0000:01:00.0:   bridge window [mem 0x0f000000-0x0f0fffff]
[    2.570112] pci 0000:01:00.0:   bridge window [mem 0x8000000000-0x80000fffff 64bit pref]
[    2.576927] pci 0000:00:01.0: PCI bridge to [bus 01-02]
[    2.577718] pci 0000:00:01.0:   bridge window [io  0x1000-0x1fff]
[    2.582524] pci 0000:00:01.0:   bridge window [mem 0x0f000000-0x0f1fffff]
[    2.588110] pci 0000:00:01.0:   bridge window [mem 0x8000000000-0x80000fffff 64bit pref]
[    2.597482] hisifb: Hisilicon Driver version: v1.0.0
[    2.598223] hisifb: no options.
[    2.599527] virtio-pci 0000:02:02.0: enabling device (0005 -> 0007)
[    2.603830] virtio-pci 0000:02:03.0: enabling device (0005 -> 0007)
[    2.611997] virtio-pci 0000:02:05.0: enabling device (0001 -> 0003)
[    2.663695] Serial: 8250/16550 driver, 4 ports, IRQ sharing enabled
[    2.665175] no indirect ISA port I/O for Hisilicon LPC uart
[    2.666447] cacheinfo: Unable to detect cache hierarchy for CPU 0
[    2.668853] loop: module loaded
[    2.669415] Loading iSCSI transport class v2.0-870.
[    2.670334] rdac: device handler registered
[    2.671030] hp_sw: device handler registered
[    2.671669] emc: device handler registered
[    2.672276] alua: device handler registered
[    2.673355] iscsi: registered transport (tcp)
[    2.675008] scsi host0: Virtio SCSI HBA
[    2.676460] hisi_sas: driver version v1.2
[    2.676555] scsi 0:0:0:0: Direct-Access     QEMU     QEMU HARDDISK    2.5+ PQ: 0 ANSI: 5
...
[   10.306380] 8021q: adding VLAN 0 to HW filter on device eth0
[   10.371687] Netfilter messages via NETLINK v0.30.
[   10.435708] ip_set: protocol 6
[  138.363437] sysrq: SysRq : Manual OOM execution
[  138.377501] kbox catch oom event.
```

```
[  138.406800] Mem-Info:
[  138.406807] active_anon:14316 inactive_anon:2337 isolated_anon:0
 active_file:9427 inactive_file:19841 isolated_file:0
 unevictable:0 dirty:4 writeback:0 unstable:0
 slab_reclaimable:4521 slab_unreclaimable:4169
 mapped:8838 shmem:2388 pagetables:313 bounce:0
 free:3294351 free_pcp:993 free_cma:0
[  138.406811] Node 0 DMA free:3005784kB min:10268kB low:12832kB high:15400kB active_anon:
11244kB inactive_anon:1432kB active_file:9200kB inactive_file:20436kB unevictable:0kB
isolated(anon):0kB isolated(file):0kB present:3145728kB managed:3063848kB mlocked:0kB dirty:
8kB writeback:0kB mapped:7992kB shmem:1512kB slab_reclaimable:2408kB slab_unreclaimable:
2600kB kernel_stack:320kB pagetables:284kB unstable:0kB bounce:0kB free_pcp:2220kB local_pcp:
684kB free_cma:0kB writeback_tmp:0kB pages_scanned:0 all_unreclaimable? no
[  138.406819] lowmem_reserve[]: 0 10130 10130
[  138.406824] Node 0 Normal free:10171620kB min:34784kB low:43480kB high:52176kB
active_anon:46020kB inactive_anon:7916kB active_file:28508kB inactive_file:58928kB
unevictable:0kB isolated(anon):0kB isolated(file):0kB present:10903552kB managed:10391500kB
mlocked:0kB dirty:8kB writeback:0kB mapped:27360kB shmem:8040kB slab_reclaimable:15676kB
slab_unreclaimable:14076kB kernel_stack:1776kB pagetables:968kB unstable:0kB bounce:0kB
free_pcp:1752kB local_pcp:212kB free_cma:0kB writeback_tmp:0kB pages_scanned:0
all_unreclaimable? no
[  138.406830] lowmem_reserve[]: 0 0 0
[  138.406833] Node 0 DMA: 152*4kB (UEM) 113*8kB (M) 83*16kB (UEM) 36*32kB (UEM) 21*64kB
(UEM) 5*128kB (UEM) 2*256kB (UM) 0*512kB 5*1024kB (UE) 2*2048kB (UR) 730*4096kB (MR) =
3005784kB
[  138.406851] Node 0 Normal: 41*4kB (UEM) 34*8kB (UEM) 41*16kB (UE) 39*32kB (UEM) 29*64kB
(UEM) 37*128kB (UEM) 22*256kB (UEM) 10*512kB (UM) 10*1024kB (U) 2*2048kB (UM) 2475*4096kB
(MR) = 10171620kB
[  138.406869] Node 0 hugepages_total=0 hugepages_free=0 hugepages_surp=0
hugepages_size=1048576kB
[  138.406872] Node 0 hugepages_total=0 hugepages_free=0 hugepages_surp=0
hugepages_size=2048kB
[  138.406873] 31654 total pagecache pages
[  138.406875] 0 pages in swap cache
[  138.406877] Swap cache stats: add 0, delete 0, find 0/0
[  138.406879] Free swap  = 6881276kB
[  138.406880] Total swap = 6881276kB
[  138.406882] 3512320 pages RAM
[  138.406883] 0 pages HighMem/MovableOnly
[  138.406885] 148483 pages reserved
[  138.406886] 0 pages cma reserved
[  138.406887] 0 pages hwpoisoned
```

3. 控制台打印日志

```
---------------area type:8, record num:1, unit_size:8 ------------
###num:0, record len:61
oom time:20160218155821-e0387
[ 2455.550908] Mem-Info:
[ 2455.553173] Node 0 DMA per-cpu:
[ 2455.556302] CPU    0: hi:    0, btch:   1 usd:   0
[ 2455.561064] CPU    1: hi:    0, btch:   1 usd:   0
[ 2455.565825] CPU    2: hi:    0, btch:   1 usd:   0
[ 2455.570586] CPU    3: hi:    0, btch:   1 usd:   0
[ 2455.575347] Node 0 DMA32 per-cpu:
[ 2455.578651] CPU    0: hi:  186, btch:  31 usd:   0
[ 2455.583412] CPU    1: hi:  186, btch:  31 usd:   0
[ 2455.588173] CPU    2: hi:  186, btch:  31 usd:   0
[ 2455.592934] CPU    3: hi:  186, btch:  31 usd:   0
oom time:20160218155821-e0387
```

1.2.4 提供系统 die/oops 信息

本节主要介绍内核发生oops时内核黑匣子所记录的异常信息。

功能概述

产品/平台业务软件或操作系统本身可能因为某种特殊原因导致访问空指针、非法访问内存空间等，触发oops事件。Kbox能够将oops事件发生的时间、发生oops进程信息、异常进程调用栈等记录到存储设备中，方便维护人员定位。

信息清单

记录的异常die/oops信息包含三个部分内容。

1. oops异常信息，最多记录信息容量为128K。该记录区主要内容包括：
   - 内核异常发生的时间(UTC时间)。
   - 当前进程的pid、进程名称。
   - 异常进程的调用轨迹及栈信息（栈信息最多打印150行）。
   - 打印系统中模块名称起始地址和长度（最多打印384条模块信息）。

     说明

     若系统中存在大量模块，且各模块的名称很长时，存在冲日志的风险。
2. 内核消息打印日志，最多记录信息容量为64K。该记录区主要内容包括：
   - 内核异常发生的时间(UTC时间)。
   - 异常发生时最近64K日志信息，即内核打印到循环缓冲区中的最后64K日志信息。
3. 控制台打印日志，最多记录信息容量为32K。该记录区主要内容包括：
   - 内核异常发生的时间(UTC时间)。
   - 其他内核打印日志消息。

     说明

     如果当前其他进程没有向控制台打印日志，收集的控制台打印日志为空。

信息举例

在日志信息文件中，包含如下内容：

1. oops异常信息

```
---------------area type:5, record num:1, unit_size:32 ------------
###num:0, record len:7365
----------KBOX_START-----------
//die事件发生的时间
die time:20180108103026-bb29
//错误类型及错误号
die info:Oops:0046
//进程的所在CPU号、进程名称
Process: bash (pid: 6111, threadinfo: ffff800336f60000, task: ffff8002ef5a3d40) on CPU: 0
//调用栈
Call Trace:
[<ffff8000005e21d0>] sysrq_handle_crash+0x28/0x38
[<ffff8000005e2e88>] __handle_sysrq+0x130/0x190
[<ffff8000005e3340>] write_sysrq_trigger+0x68/0x78
[<ffff8000002d0d20>] proc_reg_write+0x70/0xa0
[<ffff80000026023c>] __vfs_write+0x4c/0x120
[<ffff800000260c34>] vfs_write+0x9c/0x1a8
[<ffff800000261888>] SyS_write+0x68/0xd8
PC: sysrq_handle_crash+0x28/0x38
LR: __handle_sysrq+0x130/0x190
sp : ffff800336f63cf0
//寄存器信息
x29: ffff800336f63cf0 x28: ffff800336f60000
```



```
x27: ffff800000a03000 x26: 0000000000000040
x25: 000000000000011a x24: 0000000000000015
x23: 0000000000000000 x22: 0000000000000001
x21: 0000000000000063 x20: ffff80000100e000
x19: ffff8000010a1778 x18: 0000000000000000
x17: 0000ffff7e3d75b0 x16: ffff800000261820
x15: 0000000000000020 x14: 0fffffffffffffe
x13: 0000000000000010 x12: 00000000000001e6
x11: 0000000000000002 x10: 0000000000000001
x9 : ffff8000005e2e88 x8 : 0000000000000078
x7 : 0000000000000030 x6 : 0000000000000f24
x5 : 0000000000000000 x4 : 0000000000000000
x3 : 0000000000000000 x2 : cb88537fdc8ba32c
x1 : 0000000000000000 x0 : 0000000000000001

Stack:
stack grow form: ffff800336f63ce0 to ffff800336f64000
0001000000000000 000000000000000f ffff800336f63d00 ffff8000005e2e88
ffff800336f63d40 ffff8000005e3340 0000000000000002 fffffffffffffffb
ffff800336f63ea0 ffff800336f63ea0 0000000000000002 cb88537fdc8ba32c
ffff800336f63d60 ffff8000002d0d20 ffff8003352c03c0 ffff800000213704
ffff800336f63d80 ffff80000026023c ffff800000fe7000 ffff800337b65800
ffff800336f63e20 ffff800000260c34 ffff800337b65800 0000000000000002
0000ffff7e32e000 0000ffff7e32e000 ffff800336f63e20 ffff800000260d28
ffff800337b65800 0000000000000002 0000ffff7e32e000 ffff800336f63ea0
0000000000000002 0000000000000015 0000ffff7e32e000 ffff800336f63ea0
ffff800336f63e20 ffff800000260c04 ffff800337b65800 cb88537fdc8ba32c
ffff800336f63e60 ffff800000261888 ffff800000fe7000 ffff800337b65800
ffff800337b65800 0000ffff7e32e000 0000000000000002 ffff8000001768a8
0000ffffe2528830 ffff800000083f78 0000000000000200 0000ffff7e32e000
ffffffffffffffff 0000ffff7e43a438 0000000020000000 0000ffff7e32e000
...
0000ffff7e3d9e38 0000ffffe2528830 0000ffff7e43a438 0000000020000000
0000000000000000 0000000000000000 0000000000000001 0000000000000040
0000000000000000 0000000000000000 0000000000000000 0000000000000000
Code:
910003fd b90d8820 d5033e9f d2800001 (39000020)
mod_name                mod_start             core_size
ip6t_rpfilter           0xffff7ffffc368000    0x771
ipt_REJECT              0xffff7ffffc364000    0x6a7
ip6t_REJECT             0xffff7ffffc360000    0x8a6
nf_reject_ipv6          0xffff7ffffc35b000    0x1472
xt_conntrack            0xffff7ffffc357000    0xd97
ip_set                  0xffff7ffffc348000    0x8f11
......
//die发生后的动作（0：不做任何操作；1：调用panic；2：reboot）
action after die is:0
---------KBOX_END------------
```

2. 内核消息打印日志

```
message:
---------------area type:7, record num:1, unit_size:64 ------------
###num:0, record len:28148
die time:20180108103026-bb29
[    0.     0] SLUB: HWalign=64, Order=0-3, MinObjects=0, CPUs=4, Nodes=1
[    0.     0] Hierarchical RCU implementation.
[    0.     0]  Additional per-CPU info printed with stalls.
[    0.     0]  RCU restricting CPUs from NR_CPUS=128 to nr_cpu_ids=4.
[    0.     0] RCU: Adjusting geometry for rcu_fanout_leaf=16, nr_cpu_ids=4
[    0.     0] NR_IRQS:64 nr_irqs:64 0
[    0.     0] ITS: /intc/its
[    0.     0] ITS: allocated 65536 Devices @379d00000 (psz 64K, shr 1)
[    0.     0] ITS: allocated 8192 Interrupt Collections @379cb0000 (psz 64K, shr 1)
[    0.     0] GIC: using LPI property table @0x0000000379cc0000
[    0.     0] ITS: Allocated 1792 chunks for LPIs
[    0.     0] GICv3: CPU0: found redistributor 0 region 0:0x00000000080a0000
[    0.     0] CPU0: using LPI pending table @0x0000000379cd0000
[    0.     0] Architected cp15 timer(s) running at 50.00MHz (virt).
[    0.     0] clocksource arch_sys_counter: mask: 0xffffffffffffff max_cycles: 0xb8812736b,
max_idle_ns: 440795202655 ns
```

```
[    0.      1] sched_clock: 56 bits at 50MHz, resolution 20ns, wraps every 4398046511100ns
[    0.   1576] Console: colour dummy device 80x25
[    0.   2413] console [tty0] enabled
[    0.   3047] bootconsole [pl11] disabled
[    0.   3762] kmemleak: Kernel memory leak detector disabled
[    0.   3777] Calibrating delay loop (skipped), value calculated using timer frequency..
100.00 BogoMIPS (lpj=200000)
[    0.   3783] pid_max: default: 32768 minimum: 301
[    0.   3805] Security Framework initialized
[    0.  19150] Dentry cache hash table entries: 2097152 (order: 12, 16777216 bytes)
[    0.140781] Inode-cache hash table entries: 1048576 (order: 11, 8388608 bytes)
[    0.181500] Mount-cache hash table entries: 32768 (order: 6, 262144 bytes)
[    0.181537] Mountpoint-cache hash table entries: 32768 (order: 6, 262144 bytes)
[    0.194962] Initializing cgroup subsys blkio
[    0.194968] Initializing cgroup subsys memory
[    0.194979] Initializing cgroup subsys devices
[    0.194984] Initializing cgroup subsys freezer
[    0.194988] Initializing cgroup subsys net_cls
[    0.194992] Initializing cgroup subsys perf_event
[    0.194996] Initializing cgroup subsys net_prio
[    0.195001] Initializing cgroup subsys hugetlb
[    0.195012] ftrace: allocating 33040 entries in 130 pages
[    0.289970] hw perfevents: enabled with arm/armv8-pmuv3 PMU driver, 7 counters available
[    0.289987] Remapping and enabling EFI services.
[    0.289993]   EFI remap 0x0000000004000000 => 0000000040000000
[    0.290009]   EFI remap 0x0000000009010000 => 0000000044000000
...
[    9.158060] ip6_tables: (C) 2000-2006 Netfilter Core Team
[    9.331126] Ebtables v2.0 registered
[    9.629776] Netfilter messages via NETLINK v0.30.
[    9.642403] ip_set: protocol 6
[    9.670130] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[    9.670353] 8021q: adding VLAN 0 to HW filter on device eth0
[  728.614997] sysrq: SysRq : Trigger a crash
[  728.616026] Unable to handle kernel NULL pointer dereference at virtual address 00000000
[  728.616029] pgd = ffff8002efb9b000
[  728.616031] [00000000] *pgd=0000000377fb7003, *pud=000000032fafe003, *pmd=0000000000000000
[  728.616037] Internal error: Oops: 96000046 [#1] SMP
[  728.616773] kbox catch die event.
```

3. 控制台打印日志

```
console:
---------------area type:8, record num:1, unit_size:8 ------------
###num:0, record len:245
[  728.616773] kbox catch die event.
die time:20180108103026-bb29
[  728.617610] collected_len = 28119, LOG_BUF_LEN_LOCAL = 32768
[  728.749258] [kbox drv] flush data done. addr: 0xffff800358800000, size: 0x1000000
die time:20180108103026-bb29
```

1.3 如何使用 Kbox

本节主要介绍Kbox的使用流程以及使用中的注意事项。

1.3.1 使用流程

本节介绍内核黑匣子应用流程，包括修改配置文件、重启Kbox服务、查看异常信息等。

使用流程

Kbox的使用流程如图1-2所示。

图 1-2 Kbox 使用简单流程

说明

- 在EulerOS系统中，Kbox默认随内核一起发布，系统启动时Kbox服务会自动开启。
- 启动参数中必须预先配置reserve_kbox_mem=16M。若未配置或少于16M，则kbox服务启动会失败。

步骤说明

具体步骤说明如表1-2所示。

注意

1. 必须配置存储介质，否则捕获的信息不能存储，系统重启之后信息丢失。
2. 配置之后，必须重启黑匣子服务，配置内容才会生效。
3. 当hmem被设置成"on"，并且修改了hmem_param参数，必须重启kdump服务，kbox日志才能正常转储。详情请见表1-6

表 1-2 Kbox 步骤说明

编号	流程	说明
1	启动系统	在EulerOS系统中，Kbox默认随内核一起发布，系统启动时Kbox服务会自动开启。
2	修改配置文件	配置产品信息、抓取的异常事件、以及可用的存储方式。Kbox配置文件路径为/etc/kbox/config。
3	重启Kbox服务	修改配置文件后，通过重启Kbox服务，使配置文件生效。

编号	流程	说明
4	记录异常信息	当异常事件发生后，Kbox会根据配置文件，将指定的异常事件信息记录到相应的存储设备中。
5	查看异常信息	具有root权限的用户通过Kbox提供的日志导出命令，将存储设备中的异常信息导出到指定目录的文件中，进行查看。

1.3.2 修改配置文件

本节介绍如何正确配置Kbox工具。配置主要包含三方面：产品信息、异常事件和存储设备。

前提条件

- 在EulerOS系统中，Kbox正常运行。
- Kbox转储依赖的转储介质正常工作。

注意事项

用户修改完配置文件，重启Kbox后才生效。

配置步骤

以root权限，在shell命令行环境下：

步骤1 打开配置文件。在任一路径下，使用如下命令：

```
vi  /etc/kbox/config
```

配置文件内容如下：

```
# This is kbox's configuration file.
# Depending on the circumstances of each product to complete the configuration.
# * IMPORTANT *: configure parameters correctly according to the prompt message, the parameters
must contain quotes, no spaces inside and outside the quotation marks.


##############  Basic information of product  ###############
# Product name, eg: "EulerOS" "CloudEdge" "Pangea" "MSP" "N9000" "SVP" "CGP".
product="EulerOS"

# Product version number: e.g. V100R001C00B251.
version="EulerOS_V200R005C00"

# the frame number: e.g., frame = 1 ~ 9 (generally used for cascade)
frame="1"

# Slot number of the board: e.g., slot = 0 ~ 12
slot="0"

# Location board: e.g., after the front board or card (0-1)
locate="0"

# Board hardware information: e.g., pangea-v2
hard_version=""
```

```
############  Abnormal event scenarios, if the option is empty or invalid characters, the default
is no  ###############
# Panic events: mainly explicitly call the kernel panic function or abnormal triggering cause
kernel oops.
panic_event="yes"

# Normal reboot events: the event mainly triggered by user space command reboot, shutdown, halt or
init.
reboot_event="yes"

# Abnormal reboot event: the main exception is triggered by the kernel mode machine restart.
emerge_event="yes"

# Die event: the event mainly triggered by kernel mode illegal pointer accesses, etc.
die_event="yes"

# If call panic after die event process handled.
die_call_panic="no"

# Out of memory event.
oom_event="yes"

# If call panic after oom event process handled.
oom_call_panic="yes"

# R deadlock event
rlock_event="no"


###############  Storage device type and parameters  ###############
# Note: start_pddr must be a physical address, and 4K aligned with hexadecimal.
#       mem_size is a minimum of 8M bytes with hexadecimal.
# Separated by a space between the parameters.

# bios device. Used for pangea product.
bios="off"
# Pangea product' kbox stored in nvram, you can not configure this.
bios_param=""

# pcie device. Used for MSP, SVP guest etc..
pcie="off"
# Parameter for pcie device.
pcie_param="pci_bus_id=00 pci_device_id=00"

# high memory device. Used for EulerOS, CloudEdge, N9000 etc..
hmem="on"
# Parameter for high memory device.
#hmem_param="start_paddr=0x0 mem_size=0x0"
hmem_param="sym_start_paddr_name=reserve_kbox_mem_start sym_mem_size_name=reserve_kbox_mem_len"

# sigma device.
sigma="off"
sigma_param=""

############### Fault reporting interface  ###############
die_notify_func=""


################## Auto export kbox log ##################
# This option can only enable in nonvolatile storage scenes.
# If this option enable, kbox will export log to $export_kbox_log.
# Absolute path are expected and Only support for numbers, letters, and underscore.
export_log_path=""
```

步骤2 修改配置文件。

按照当前系统的硬件环境，配置好系统的基本信息、抓取异常的事件以及存储设备。

注意

1. 参数值请严格按照说明进行配置，配置参数必须使用英文引号，引号内不能有空格。如果字符中间需要空格，请以"_"字符代替。配置为yes/no或YES/NO、on/off或ON/OFF的配置项，大小写要统一，且不能包含空格，否则默认处理为no。
2. 异常事件必须至少配置1项，否则Kbox不能加载。必须配置存储设备，否则异常信息不能转储，Kbox重启后信息丢失。
3. 修改配置文件后，请重启Kbox，否则配置不生效。

具体参数说明和配置指导如表1-3、表1-4和表1-5所示：

表 1-3 产品信息

列序号	参数名称	说明	参数值	是否必配
1	product	当前设备名称（或产品名称）。	用户自定义字符串。	否
2	version	当前系统发布的版本号。	用户自定义字符串。	否
3	frame	当前机架号。	大于0的整数。	否
4	slot	当前单板槽位号。	大于0的整数。	否
5	locate	单板位置。	0或1。	否
6	hard_version	硬件版本。	用户自定义字符串。	否

表 1-4 异常事件

列序号	参数名称	说明	参数值	是否支持	是否必配
1	panic_event	panic事件。	yes/no	是	是，至少配置一种异常事件。
2	reboot_event	安全重启事件。	yes/no	是	
3	emerge_event	异常重启事件。	yes/no	是	
4	die_event	die事件。	yes/no	是	
5	rlock_event	R死锁事件。	yes/no	否	
6	oom_event	oom事件。	yes/no	是	

列序号	参数名称	说明	参数值	是否支持	是否必配
7	oom_call_panic	oom发生后是否调用panic。	yes/no	是	否
8	die_call_panic	die事件后是否调用panic	yes/no	是	

说明

只有在oom_event项设置为yes时，oom_call_panic配置项才能够生效。oom_call_panic默认设置为yes，如果系统需要在oom时，不让kbox调用panic复位，请将oom_call_panic设置为no，这种情况下kbox不再抓取oom事件日志，只打印内存占用高的top10进程信息。

表 1-5 存储设备配置

列序号	参数名称	说明	参数值	是否必配
1	bios	bios nvram存储。	on/off	是，至少配置其中一种存储方式。
2	pcie	PCIE存储。	on/off	
3	hmem	普通内存存储或者高端内存存储。	on/off	

说明

1. 表 3和表4表 4必须结合使用。
2. EulerOS默认配置hmem="on"。
3. 通用服务器默认使用hmem参数。

表 1-6 存储配置参数

列序号	参数名称	说明	参数值	是否必配
1	bios_param	biosnvram驱动参数	驱动自带默认参数	否
2	pcie_param	pcie驱动参数	pci_bus_id=00 pci_device_id=00	是

列序号	参数名称	说明	参数值	是否必配
3	hmem_param	保留内存驱动参数	该参数有两种取值方式： ● sym_start_paddr_name=reserve_kbox_mem_start sym_mem_size_name=reserve_kbox_mem_len ● start_paddr=0xxxxxx mem_size=0xxxxx。start_paddr和mem_size为16进制整数，start_paddr为起始物理地址，mem_size为内存大小。 **说明** ● 当前默认为第一种配置方式，建议用户不要修改。 ● 当hmem被设置成"on"，并且修改了hmem_param参数时，必须**重启Kdump服务...**，kbox日志才能正常转储。因为hmem类型的存储设备是利用kdump机制转储kbox日志的。	是

注意

如果没有非易失性存储设备，可以通过启动参数保留内存，用普通内存代替非易失性存储介质，然后通过kdump将这段内存保存为内存镜像，该内存镜像可通过**export_kbox_img_to_txt**命令读取。

表 1-7 导出 kbox 日志配置参数

序列号	参数名称	说明	参数值	是否必配
1	export_log_path	在有非遗失存储的服务器上导出上一次复位日志到指定的路径。仅支持数字/字母/下划线组成的绝对路径。如果配置为空，表示不导出。	日志绝对路径	否

----结束

1.3.3 重启 Kbox 服务

本节主要介绍Kbox运行条件和加载步骤。

前提条件

- 在EulerOS系统中，Kbox正常运行。
- Kbox转储依赖的转储介质正常工作。

加载步骤

在正常启动的EulerOS系统中，以root权限登录系统。已经按照要求，完成配置文件的修改。

在shell命令行环境中，在任意路径下，使用命令：

```
#systemctl  restart kbox
```

验证加载是否成功

说明

系统关机和重启，Kbox服务不停止。系统关机和重启调用systemctl stop kbox命令，不会停止Kbox服务，该接口为空。

查看Kbox运行状态。使用命令：

```
#systemctl  status   kbox
```

- 如果显示active，则表示Kbox模块加载成功，并且已经在运行。
- 如果显示不是active，则表示Kbox模块未加载或未加载成功。请仔细检查Kbox配置文件配置参数，确保无误后重新加载。更多信息请参见**1.5 常见问题处理**。

1.3.4 查看异常信息

本节介绍如何查看保存在存储设备中的异常信息。

前提条件

- 在EulerOS系统中，Kbox正常运行。
- Kbox转储依赖的转储介质正常工作。
- 具有root权限的用户才能正常查看Kbox相关信息。

导出异常信息

系统启动时，Kbox日志被保存到/var/crash(默认位置，kdump配置文件中path参数可以重新指定位置)，每次系统异常的日志都保存到以“ip-时间点”命名的目录中，该目录下的xxx-xxx.img或者kbox_log.txt就是kbox日志。

说明

当前kbox日志存放的目录为“127.0.0.1-时间点”，127.0.0.1是指该kbox日志仅存放在本机，不会传输至其他服务器。

```
[root@localhost 127.0.0.1-2018.01.26-09:39:27]# ls
13f000000-1000000.img.gz  kbox_log.txt  kdump_status.log  vmcore-dmesg.txt
```

kbox_log.txt为解析好的日志，也可以使用gunzip命令解压xxx.img.gz后，如下命令对日志镜像解压解析：

```
# export_kbox_img_to_txt  -i 13f000000-1000000.img
```


提示export_kbox_img_to_txt: parse kbox data into /var/log/kbox/kbox_log.txt success.

kbox_log.txt即为解析出来的kbox日志

用户可使用vi命令查看异常信息的日志文件。此处以OOM信息为例。详细信息说明请参见提供系统oom信息。

```
area type: 0-panic 1-reboot 2-emerge 3-intermit 4-oom 5-oops/die 6-rlock 7-message 8-console
---------------area type:4, record num:1, unit_size:64 ------------


###num:0, record len:34586
---------KBOX_START----------
oom time:20170310212837-d6cf8
Current Pid: 46, comm: kworker/3:1 Tainted: G          OE   4.1.44-03.35.vhulk1711.1.1.aarch64 #1
Call Trace:
 [<ffffffffa036eb4b>] ? kbox_show_cur_stack+0x6b/0x80 [kbox]
 [<ffffffffa036f4e3>] ? kbox_dump_oom+0x13/0x30 [kbox]
 [<ffffffffa036e630>] ? kbox_oom_callback+0xe0/0x220 [kbox]
 [<ffffffff811ddb00>] ? pollwake+0x20/0x90
 [<ffffffff810aa700>] ? wake_up_state+0x20/0x20
 [<ffffffff810a1958>] ? __wake_up_common+0x58/0x90
 [<ffffffff81612b8c>] ? notifier_call_chain+0x4c/0x70
 [<ffffffff8109e81d>] ? __blocking_notifier_call_chain+0x4d/0x70
 [<ffffffff8109e856>] ? blocking_notifier_call_chain+0x16/0x20
 [<ffffffff8115d29e>] ? out_of_memory+0x4e/0x4f0
 [<ffffffff813976ac>] ? tty_ldisc_deref+0x3c/0xa0
 [<ffffffff8139bb8d>] ? moom_callback+0x4d/0x50
 [<ffffffff8109015b>] ? process_one_work+0x17b/0x470
 [<ffffffff81090f2b>] ? worker_thread+0x11b/0x400
 [<ffffffff81090e10>] ? rescuer_thread+0x400/0x400
 [<ffffffff8109830f>] ? kthread+0xcf/0xe0
 [<ffffffff81098240>] ? kthread_create_on_node+0x140/0x140
 [<ffffffff816170d8>] ? ret_from_fork+0x58/0x90
 [<ffffffff81098240>] ? kthread_create_on_node+0x140/0x140
Stack:
 ffff88013861bba0 ffff880138582e00 0000000000000000 0000000000000000
 ffff88013861bdc8 0000000000000000 ffff88013861bc08 ffffffffa0375091
 000000042cc68d17 0000000000000000 ffff880138582e00 ffff88013861bbf8
 0000000000000014 ffff88013861bc40 ffffffffa036eabd ffff880100000004
 ffffffffa036eb4b 000000002cc68d17 0000000000000000 ffff88013861bc64
 ffff88013861bc50 ffffffffa036f4e3 ffff88013861bce8 ffffffffa036e630
 363130323861bc70 3832313230313330 38666336642d3733 ffffffff811ddb00
 0000000000000000 ffff8800362f1700 ffffffff810aa700 0000000000000000
 0000000000000000 000000002cc68d17 ffff88013861bcf8 ffffffff810a1958
 0000000100000000 000000002cc68d17 00000000ffffffff 0000000000000000
 0000000000000000 ffff88013861bd20 ffffffff81612b8c ffffffff8197fd00
 0000000000000000 0000000000000000 ffff88013861bdc8 00000000ffffffff
 ffff88013861bd60 ffffffff8109e81d 0000000000000000 0000000000000000
 0000000000000001 ffff88013ed92ec0 ffff88013efd8000 00000000000000d0
 ffff88013861bd70 ffffffff8109e856 ffff88013861be08 ffffffff8115d29e
 ffff88013ed93680 ffff88013ed93680 ffff88013861bde0 0000000000000286
 0000000000000282 ffff880036acd980 ffff88013861bdd0 ffffffff813976ac
 0000000000000282 0000000000000000 ffff88013861be18 000000002cc68d17
 ffffffff819c57e0 ffff8801385ea000 ffff88013ed92ec0 ffff88013ed96f00
 00000000000000c0 ffff88013861be18 ffffffff8139bb8d ffff88013861be60
 ffffffff8109015b 000000003ed92ed8 0000000000000000 ffff88013ed92ed8
 ffff8801385ea030 ffff880138582e00 ffff8801385ea000 ffff88013ed92ec0
 ffff88013861bec0 ffffffff81090f2b ffff88013861bfd8 ffff88013861bfd8
 ffff880138582e00 ffff880138582e00 ffff880138582e00 ffff880138417d38
```

1.4 查看 Kbox 相关信息

本节介绍如何查看Kbox相关的信息。

前提条件

- 在EulerOS系统中，Kbox正常运行。
- Kbox转储依赖的转储介质正常工作。

查看操作

- 查看Kbox的运行状态。使用命令：

```
systemctl  status  kbox
kbox.service - Crash message collector   Loaded: loaded (/usr/lib/systemd/system/kbox.service;
disabled)   Active: active (exited) since Thu 2015-11-26 10:39:47 EST; 13s ago
  Process: 26545 ExecStart=/usr/sbin/kbox start (code=exited, status=0/SUCCESS)
 Main PID: 26545 (code=exited, status=0/SUCCESS)

Nov 26 10:39:46 localhost.localdomain kbox[26545]: kbox: kbox module has been loaded successfully!
Nov 26 10:39:46 localhost.localdomain kbox[26545]: driver: drivers have been loaded successfully!
Nov 26 10:39:47 localhost.localdomain systemd[1]: Started Crash message collector.
```

- 显示
  - 如果Kbox正常运行，显示active状态。如下图所示：

    ```
    kbox.service - Crash message collector
       Loaded: loaded (/usr/lib/systemd/system/kbox.service; disabled)
       Active: active (exited) since Thu 2015-11-26 10:39:47 EST; 13s ago
    ```

  - 如果显示failed状态，则表示Kbox未加载到系统中。如下图所示：

    ```
    kbox.service - Crash message collector
       Loaded: loaded (/usr/lib/systemd/system/kbox.service; disabled)
       Active: failed (Result: exit-code) since Thu 2015-11-26 10:44:43 EST; 5s ago
      Process: 29347 ExecStart=/usr/sbin/kbox start (code=exited, status=1/FAILURE)
     Main PID: 29347 (code=exited, status=1/FAILURE)
    ```

- 查看当前系统中的临时存储区(region)。

  Kbox启动后，默认会创建多个临时存储区，保存对应类型的日志信息。查看当前系统中已有的临时存储区命令如下

```
the effective info in /etc/kbox/config:

------------- product      info -------------
product       name: euler
version     number: V200R002C10
frame       number: 1
slot        number: 0
locate        info: 0

------------- enabled    events -------------
panic_event    die_event      oom_event
oom_call_panic


------------- enabled  storages -------------
hmem


the regions and storages list:

------------- regions      list -------------
console die message oom panic

------------- storages     list -------------
hmem
```

  临时存储区域说明如**表1-8**所示。

表 1-8 存储区域说明

存储区域名称	说明
console	用来存储异常发生后，打印到控制台的日志信息，如果信息过多，会产生回绕，覆盖最先保存的信息。该临时存储区大小为32K byte。
die	用来存储由于内核态进程发生oops(非法访问内存空间、使用空指针等)事件所抓取的异常信息。该临时存储区大小为128K byte。
message	用来存储异常发生时，最近的64K byte内核打印信息。
oom	用来存放oom事件抓取的信息，该临时存储区大小为256K byte。
panic	用来存放panic事件发生时，抓取的异常信息。该临时存储区大小为32K byte。
rlock	用来存放发生rlock(内核软狗和硬狗狗叫)时抓取到的异常信息，该临时存储区大小为256K byte。

- 查看存储设备的全局信息

  显示当前系统中注册的存储设备

  所谓的存储设备就是注册到Kbox模块中的存储介质，包括NVRAM、PCIE设备内存、BMC设备内存等。当系统发生异常后，Kbox将临时存储区的日志信息刷新到这些设备中。目前默认存储设备为hmem(普通内存)。使用举例如下：


```
[root@localhost home]# kboxstatus
the effective info in /etc/kbox/config:

------------ product      info ------------
product       name: EulerOS
version     number: EulerOS_V200R005C00
frame       number: 1
slot        number: 0
locate        info: 0

------------ enabled    events ------------
panic_event    reboot_event    emerge_event
die_event      oom_event       oom_call_panic


------------ enabled  storages ------------
hmem


the regions and storages list:

------------ regions      list ------------
console die emerge message oom panic
reboot

------------ storages     list ------------
hmem
```

1.5 常见问题处理

本节介绍使用Kbox常见的问题以及处理方法。

Kbox 导出日志失败

1. 通过命令systemctl status kbox，检查Kbox是否启动。如果没有启动，运行systemctl start kbox命令启动服务，启动失败，请检查kbox配置是否正确。
2. 通过命令systemctl status kdump，检查kdump服务是否启动正常。如果没有正常启动，运行systemctl start kdump命令启动kdump服务，如果启动失败，请检查kdump配置是否正确。

   说明

   kbox日志都是通过kdump保存。

Kbox 不能抓取异常信息

通过命令systemctl status kbox，确认Kbox服务是否正常启动，如果没启动服务，运行systemctl start kbox命令启动服务。

Kbox 服务启动失败

请按照如下步骤逐一排查：

1. 确定当前操作系统是否为EulerOS，使用命令“cat /etc/os-release”进行查询。Kbox当前仅支持EulerOS版本，在其它版本的Linux操作系统上无法使用。
2. 通过命令systemctl status kbox和journalctl -l，确认Kbox服务启动失败提示信息。
3. Kbox配置文件修改有误。请核对修改的配置项，确保修改正确。
4. 预留内存出现错误也会导致kbox启动失败。

   系统在启动阶段会为kbox调测工具预留分配16M的内存。这是通过在grub配置文件的cmdline参数中增加reserve_kbox_mem=16M来实现。

   内存预留算法分两步：

   a. 在e820图中，从高端地址到低端地址遍历每个usable状态的内存段，直到有能满足16MB大小段为止。

   b. 在找到的内存段中再从高端地址向低端地址reserved 16MB内存。

   注意事项：

   （1）如果系统可用内存小于16M，则kbox调测工具启动会失败。

   （2）使用memmap预留内存需要注意避免跟reserve_kbox_mem内存段冲突，memmap在预留内存时不判断内存段状态。这种强制分配的方式可能会对已经reserved的内存段再次reserved而造成冲突。

   下图所示，正常情况下reserve_kbox_mem预留内存段前后的地址段范围和内存状态的改变。

```
[root@localhost home]# cat /proc/iomem | grep Kbox
  398800000-3997fffff : Kbox reserved
```

关闭 kbox 捕获 rlock 场景

在EulerOS系统中，kbox默认未配置捕获rlock事件，系统rlock后调用panic复位。如果产品需要在死锁场景不复位系统，首先要去掉启动参数中softlockup_panic=1参数，并设置/etc/kbox/config文件中rlock_event="no"，然后重启系统生效。

关闭 kbox 捕获 oom 和 panic

在EulerOS系统中，kbox默认配置捕获oom事件，并调用panic。如果产品需要配置oom后继续运行，需要关闭kbox捕获oom和调用panic，设置/etc/kbox/config文件中oom_call_panic="no"。

1.6 附录

1.6.1 命令参考

说明

/usr/sbin/kbox命令为kbox内部使用命令，用来启动kbox，请用户不要使用。

启动 Kbox 服务

```
systemctl  start  kbox
```

重启 Kbox 服务

```
systemctl  restart  kbox
```

查看 Kbox 状态

```
systemctl  status  kbox
```

查看 Kbox 有效配置信息、临时存储区名称及已注册的存储设备

```
kboxstatus
```

说明

如果kbox服务未启动，则不会显示当前临时存储区(region)以及注册的存储设备

导出日志信息

os_kbox_config命令用来导出非易失性存储里面的日志。

说明

如果系统中没有非易失性存储设备，则执行此命令无效。

os_kbox_config -o dev=[存储设备名称]，type=[导出日志类型]，index=[第几条] | -h | -v

详细的参数说明请参见**表1-9**。

表 1-9 参数说明

参数名称	描述
dev=[存储设备名称]	当前系统中注册的存储设备名称（biosnvram、hmem、pcie），保存异常信息。
type=[导出日志类型]	导出日志类型，与region名称相同，可取oom/console/message/die/panic/rlock。
index=[第几条]	查看第几条信息，取值[1,32]。取1时为最近的一条记录，最多记录32条信息。
-h	查看帮助。
-v	查看kbox版本信息。

解析日志文件

由于通用服务器没有非易失性存储系统，因此kbox只能配置为在普通内存中存储日志。当系统crash时，kdump会保存这段普通内存为内存镜像文件，通过export_kbox_img_to_txt来解析这个镜像文件，输出文本格式的日志文件。

export_kbox_img_to_txt -i kbox文件

说明

- kbox文件默认使用gzip压缩，解析时请先解压，参考命令gunzip xxx-xxx.img.gz。
- 该命令只适用于通用服务器上kbox日志解析。

2 kdump

Kdump是一种内核崩溃转储机制，在系统崩溃时收集整个内存信息，用于异常事件的定位分析。

2.1 特性描述

当程序或操作人员通过魔术键c或其他原因导致标准内核（即正常运行的内核）崩溃时，Kdump会切换到crash内核，该内核用于捕获标准内核崩溃时的内存信息。

注意

使用默认配置，Kdump在系统崩溃时是不会保存用户数据的。如果用户修改相关配置，可能会保存部分用户数据，请谨慎使用。如需关闭kdump功能，请参考**2.4 管理kdump服务**。

2.2 约束限制

- 当前仅支持EulerOS操作系统。

  说明

  XEN平台不支持kdump。目前kdump只在物理机及uvp kvm虚拟化平台上支持。

- kdump过程受到磁盘写速度、内存使用率、逻辑狗叫时间等因素限制，kdump过程产生的vmcore可能存在不完整的情况。
- kdump时间与内存规格强相关，内存越大，dump时间越长，产生的vmcore越大。

- 不支持通过网络保存vmcore（如NFS或者网络盘）。
- kdump配置文件修改后，必须重启kdump服务。如果系统时间发生跳变（跳变成比修改时间早的时间），重启服务检测不到配置文件有修改，修改不会生效。
- 由于3108 raid卡硬件问题，在使用了3108 raid卡的EulerOS arm64系统中不支持kdump特性。
- 创建虚拟机时，必须在虚拟机配置文件中配置

```
<features>
<acpi/>
</features>
```

否则虚拟机中kdump功能不可用。

2.3 配置 kdump 参数

获取 Kdump 信息

说明

系统启动后，Kdump服务默认处于工作状态。

每发生一次内核崩溃，Kdump便会在/var/crash目录(默认)下生成一个以127.0.0.1-时间戳命名的文件夹。/var/crash目录下默认只能存放1个文件夹，即上一次内核崩溃的Kdump信息。当再次发生内核crash时，则会先删除之前保存的目录，再保存本次crash的信息。

默认情况下，kdump保存了4个文件：

- xxx-xxx.img.gz

  该文件为kbox存储日志元数据文件，解压后使用如下命令解析。

  ```
  export_kbox_img_to_txt -i xxx-xxx.img  //解析
  ```

  解析后的文件默认存放在/var/log/kbox/kbox_log.txt，查看该文件即可。
- kbox_log.txt

  kbox元数据解析后的文本文件，即kbox捕获的日志。
- kdump_status.log

  kdump转储过程日志。
- vmcore-dmesg.txt

  该文件是内核崩溃时的dmesg信息。

配置 Kdump 参数

注意

- Kdump配置文件用户使用默认配置即可，随意修改可能导致异常。
- 系统启动后的cmdline参数中必须包含保留内核参数crashkernel=1024M（物理机环境）或者crashkernel=256M（虚拟机），不要随意修改(用户可通过命令“cat /proc/cmdline”进行查看)。1024M（或256M）是预留给kdump内核使用的内存大小。
- kdump配置文件修改后，必须重启kdump服务。如果系统时间发生跳变（跳变成比修改时间早的时间），重启服务检测不到配置文件有修改，修改不会生效。解决办法：删除/boot/initramfs-4.1.x-xxx-xxxx.aarch64kdump.img文件后，重启kdump服务。

Kdump各参数在“/etc/kdump.conf ”文件中配置，这里介绍几个重要的参数配置，如表2-1。

表 2-1 Kdump 参数说明

参数	含义	取值
default	kdump失败后的默认操作	● shell：kdump失败后启动bash。 ● reboot | halt | poweroff：重启或者关机。 当前默认设置为reboot。
path	保存kdump日志的绝对路径	默认保存在/var/crash路径下，如果配置其他路径，路径必须存在。 **说明** 使用默认目录或者规划独立的目录，不要与其他日志目录混用，否则存在删除非kdump日志的风险。
dracut_args	dracat打包kdump initrd的参数	默认为dracut_args --omit-drivers "mlx4_core mlx4_en" --omit "ramdisk network ifcfg qemu-net" --install "chmod findmnt du dump_mem_imag export_kbox_img_to_txt awk euleros_set_reboot_timer.sh" --nofscks，et_reboot_timer.sh" --nofscks，该参数较长，以实际为准。用户可了解dracat功能后设置。如果用户在EulerOS系统中加入了自研的网络驱动或者依赖了操作系统的网络驱动，必须将新增的驱动加入到--omit-driver列表中。
core_collector	保存kdump日志工具	默认为makedumpfile，其中-d项指定过滤内存级别为1，用户无需修改。
kdump_obj	设置kdump保存kbox日志或者vmcore	● vmcore：保存vmcore和用户配置的内存段。 ● kbox：保存kbox。 ● all：保存vmcore、用户配置内存段和kbox，默认。
kbox_mem	kbox日志在内存中的存放区域	kbox自填，用户慎用。

参数	含义	取值
keep_old_dumps	历史kdump日志保存个数	● 0和-1：只保存当前日志，默认设置-1。 ● 1 | 2 | 3 | ...：支持设置>0个历史日志。 ● 其他配置不支持。
watchdog_time	设置kdump过程自动复位时间（秒）	默认为3600秒，如果kdump_obj配置项设置为vmcore或者all，该项设置为300秒。
kdump_post	kdump流程中当日志转储完成后执行的脚本	默认配置请不要修改。
kdump_pre	kdump流程中当日志转储开始前执行的脚本	默认配置请不要修改。
vmcore_size_limit	dump目录使用的磁盘大小限制（MB）	默认是0，表示不限制。 **说明** ● 单位是MB。 ● 设置的最大值不能超过kdump的日志转储目录所在分区的磁盘大小。 例如此参数值设置为1024，则表示日志转储目录的磁盘使用量如果大于1G，那么kdump会删除vmcore文件，避免转储目录所在的分区因为磁盘占用过大而导致系统异常。
extra_dump_mem	kdump保存用户预留内存	默认不保存。用户可配置十六进制起始物理地址-十六进制长度，多段内存之间，使用分号隔开。如：0x10000-0x1000;0x40000-0x2000。 **注意** 起始地址和长度必须页对齐。
mem_compress_type	用户预留内存保存时使用的压缩格式	默认为gzip，当前只支持gzip。

2.4 管理 kdump 服务

以下命令用于启动、停止或查询Kdump服务：

- 停止Kdump服务
```
#systemctl  stop  kdump
```


- 启动Kdump服务
```
#systemctl  start  kdump
```
- 重启Kdump服务
```
#systemctl  restart  kdump
```
- 查询Kdump服务
```
#systemctl  status  kdump
```

2.5 使用 crash 工具解析 vmcore 文件

用户可使用EulerOS提供的crash工具，对Kdump保存的vmcore文件进行分析，进而定位问题。工具使用流程如下：

1. 在CMC发布包的“Software\aarch64\EulerOS-V2.0SP3-aarch64-dvd.iso”镜像中，获取crash工具crash-7.*.aarch64.rpm和内核调试文件kernel-debuginfo-4.1.*.aarch64.rpm。
2. 在EulerOS环境下，将获取的rpm文件解压到任意目录，提取crash和vmlinux文件。
3. 使用crash命令开始解析：
```
./crash ./vmlinux  /var/crash/127.*/vmcore
```
打印系统的相关信息，如内核、CPU、内存、PANIC信息等（内核版本信息因版本不同，有差异）：
```
      KERNEL: ./vmlinux
    DUMPFILE: /var/crash/127.0.0.1-2018.01.08-06:26:36/vmcore  [PARTIAL DUMP]
        CPUS: 4
        DATE: Mon Jan  8 06:26:12 2018
      UPTIME: 00:37:53
LOAD AVERAGE: 0.28, 0.17, 0.07
       TASKS: 128
    NODENAME: localhost.localdomain
     RELEASE: 4.1.44-03.35.vhulk1711.1.1.aarch64
     VERSION: #1 SMP Fri Jan 5 08:25:32 UTC 2018
     MACHINE: aarch64  (unknown Mhz)
      MEMORY: 13.4 GB
       PANIC: "sysrq: SysRq : Trigger a crash"
         PID: 6127
     COMMAND: "bash"
        TASK: ffff8000ba814980  [THREAD_INFO: ffff800337dc4000]
         CPU: 3
       STATE: TASK_RUNNING (SYSRQ)
crash>
```
EulerOS提供的crash工具提供多种命令，帮助用户查看堆栈、日志、内存等信息，使用“help”查看crash支持的命令：
```
crash> help
*             files         mach          repeat        timer ascii
fuser         mount         search        vm    bt            gdb
net           set           vtop  btop          help          p
sig           waitq dev           ipcs          ps            struct
whatisdis           irq           pte           swap          wr    eval
kmem          ptob          sym           q     exit          list
ptov          sys
extend        log           rd            task
```

说明

如果循环缓冲区日志被重定向到nvram，则log命令不可用。Nvram内存段目前不能保存到vmcore中。

各命令的具体说明可通过“man xxx”或者“help xxx”进行查看。例如：
```
crash> man bt
NAME
  bt - backtrace
```



```
SYNOPSIS
  bt [-a|-g|-r|-t|-T|-l|-e|-E|-f|-F|-o|-O] [-R ref] [-I ip] [-S sp] [pid | task]
DESCRIPTION
  Display a kernel stack backtrace.  If no arguments are given, the stack
  trace of the current context will be displayed.
       -a  displays the stack traces of the active task on each CPU.
           (only applicable to crash dumps)
       -g  displays the stack traces of all threads in the thread group of
           the target task; the thread group leader will be displayed first.
       -r  display raw stack data, consisting of a memory dump of the two
           pages of memory containing the task_union structure.
……
```

2.6 常见问题分析方法

踩堆栈问题

1. 编写代码模拟stack被踩。

```
#include <linux/kernel.h>
#include <linux/module.h>


static int __init a_init(void)
{
        char msg[10] = {0};

        printk("a init\n");

        strcpy(msg, "this modules is testing module,
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaa");


        return 0;
}

static void __exit a_exit(void)
{
        printk("a exit\n");
}

module_init(a_init)
module_exit(a_exit)
MODULE_LICENSE("GPL");
```

2. 分析方法（aarch64架构下分析方法相同，代码略有差异）：
   使用“bt”命令显示出问题进程堆栈，发现无法解析堆栈。

```
crash> bt
PID: 13360  TASK: ffff8801e386c140  CPU: 2   COMMAND: "insmod"
 #0 [ffff8801fddabc60] machine_kexec at ffffffff8101fc9b
 #1 [ffff8801fddabcb0] crash_kexec at ffffffff81082115
 #2 [ffff8801fddabd80] show_registers at ffffffff81006979
 #3 [ffff8801fddabde0] __die at ffffffff8138e3e7
 #4 [ffff8801fddabe10] die at ffffffff81007623
 #5 [ffff8801fddabe40] do_general_protection at ffffffff8138e0a9
 #6 [ffff8801fddabe70] general_protection at ffffffff8138d7cf
    [exception RIP: init_module+60]
    RIP: ffffffffa087e03c  RSP: ffff8801fddabf20  RFLAGS: 00010246
    RAX: 0000000000000000  RBX: 0000000000614010  RCX: 0000000000000000
    RDX: 0000000000000007  RSI: ffffffffa0a4c117  RDI: ffff8801fddabfe7
    RBP: 20676e6974736574   R8: 0000000000000000   R9: 0000000000000400
    R10: 0000000000000000  R11: 0000000000000000  R12: ffffffffa087e000
    R13: 0000000000000000  R14: 00007f139d2f6000  R15: 0000000000000002
    ORIG_RAX: ffffffffffffffff  CS: 0010  SS: 0018
    RIP: 6161616161616161  RSP: 00007fff37c795f8  RFLAGS: 00010202
    RAX: 6161616161616161  RBX: 6161616161616161  RCX: 6161616161616161
    RDX: 6161616161616161  RSI: 6161616161616161  RDI: 6161616161616161
    RBP: 6161616161616161   R8: 6161616161616161   R9: 6161616161616161
    R10: 6161616161616161  R11: 6161616161616161  R12: 6161616161616161
    R13: 6161616161616161  R14: 6161616161616161  R15: 6161616161616161
    ORIG_RAX: 6161616161616161  CS: 616161616161  SS: 002b
bt: WARNING: possibly bogus exception frame
```

根据RSP: ffff8801fddabf20，读取stack的内存，根据内存内容规律推出代码位置。

```
crash> rd ffff8801fddabe00 100
ffff8801fddabe00:  0000000000000292 ffff8801fddabe38   ........8.......
ffff8801fddabe10:  ffffffff81007623 ffff8801e386c140   #v......@.......
ffff8801fddabe20:  ffff8801fddabe78 0000000000000000   x...............
ffff8801fddabe30:  00007f139d2f6000 ffff8801fddabe68   .`/.....h.......
ffff8801fddabe40:  ffffffff8138e0a9 ffffffff8156f610   ..8.......V.....
ffff8801fddabe50:  0000000000000001 ffffffffa087e000   ................
ffff8801fddabe60:  0000000000000000 20676e6974736574   ........testing
ffff8801fddabe70:  ffffffff8138d7cf 0000000000000002   ..8.............
ffff8801fddabe80:  00007f139d2f6000 0000000000000000   .`/.............
ffff8801fddabe90:  ffffffffa087e000 20676e6974736574   ........testing
ffff8801fddabea0:  0000000000614010 0000000000000000   .@a.............
ffff8801fddabeb0:  0000000000000000 0000000000000400   ................
ffff8801fddabec0:  0000000000000000 0000000000000000   ................
ffff8801fddabed0:  0000000000000000 0000000000000007   ................
ffff8801fddabee0:  ffffffffa0a4c117 ffff8801fddabfe7   ................
ffff8801fddabef0:  ffffffffffffffff ffffffffa087e03c   ........<.......
ffff8801fddabf00:  0000000000000010 0000000000010246   ........F.......
ffff8801fddabf10:  ffff8801fddabf20 0000000000000018    ...............
ffff8801fddabf20:  202c656c75646f6d 6161616161616161   module, aaaaaaaa
ffff8801fddabf30:  6161616161616161 6161616161616161   aaaaaaaaaaaaaaaa
ffff8801fddabf40:  6161616161616161 6161616161616161   aaaaaaaaaaaaaaaa
ffff8801fddabf50:  6161616161616161 6161616161616161   aaaaaaaaaaaaaaaa
ffff8801fddabf60:  6161616161616161 6161616161616161   aaaaaaaaaaaaaaaa
ffff8801fddabf70:  6161616161616161 6161616161616161   aaaaaaaaaaaaaaaa
ffff8801fddabf80:  6161616161616161 6161616161616161   aaaaaaaaaaaaaaaa
ffff8801fddabf90:  6161616161616161 6161616161616161   aaaaaaaaaaaaaaaa
ffff8801fddabfa0:  6161616161616161 6161616161616161   aaaaaaaaaaaaaaaa
ffff8801fddabfb0:  6161616161616161 6161616161616161   aaaaaaaaaaaaaaaa
ffff8801fddabfc0:  6161616161616161 6161616161616161   aaaaaaaaaaaaaaaa
ffff8801fddabfd0:  6161616161616161 6161616161616161   aaaaaaaaaaaaaaaa
ffff8801fddabfe0:  0000616161616161 0000000000010202   aaaaaa..........
ffff8801fddabff0:  00007fff37c795f8 000000000000002b   ...7....+.......
```

踩页表问题

1. 编写代码模拟页表被踩。

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/kprobes.h>

#include <linux/kallsyms.h>
#include <linux/syscalls.h>
#include <linux/slab.h>
#include <linux/kdebug.h>
#include <asm/apic.h>
#include <asm/pgalloc.h>

static int __init jprobe_init(void)
{
        unsigned long address;
        pgd_t *pgd;
        pud_t *pud;
        pmd_t *pmd;

        printk(KERN_INFO "zk--- in \n");

        address = (unsigned long)kmalloc(512 ,GFP_KERNEL);
        pgd = pgd_offset(current->active_mm, address);
        pud = pud_offset(pgd, address);
        pmd = pmd_offset(pud, address);

        memset(pmd, 0, 16);

        printk("test: %x \n", *(int*)address);

        return 0;
}

static void __exit jprobe_exit(void)
{
        printk(KERN_INFO "zk--- out \n");
}

module_init(jprobe_init)
module_exit(jprobe_exit)
MODULE_LICENSE("GPL");
```

2. 分析方法：

   观察出错的堆栈，可以发现是一个地址错误。


```
crash> bt
PID: 13016  TASK: ffff88003595c080  CPU: 0   COMMAND: "insmod"
 #0 [ffff88008b0f7b10] machine_kexec at ffffffff81021a54
 #1 [ffff88008b0f7b60] crash_kexec at ffffffff8108baf8
 #2 [ffff88008b0f7c30] show_registers at ffffffff81006bf8
 #3 [ffff88008b0f7c90] __die at ffffffff81400f2e
 #4 [ffff88008b0f7cc0] no_context at ffffffff8102f733
 #5 [ffff88008b0f7d00] __bad_area_nosemaphore at ffffffff8102f925
 #6 [ffff88008b0f7dd0] bad_area_nosemaphore at ffffffff8102f9fe
 #7 [ffff88008b0f7de0] do_page_fault at ffffffff81402b0e
 #8 [ffff88008b0f7e30] page_fault at ffffffff8140029f
    [exception RIP: acct_collect+88]
    RIP: ffffffff8108af68  RSP: ffff88008b0f7ee8  RFLAGS: 00010286
    RAX: ffff8800658ba6b8  RBX: 0000000000002000  RCX: ffff88006b050480
    RDX: 0000000000000000  RSI: 0000000000000015  RDI: ffffffff81732a20
    RBP: ffff88008b0f7f08   R8: 0000000000000000   R9: fffffffffffffa8
    R10: 00007f7c07212700  R11: 0000000000000246  R12: ffff8800346ac400
    R13: 0000000000000000  R14: 0000000000000000  R15: 00007f7c071d0010
    ORIG_RAX: ffffffffffffffff  CS: 0010  SS: 0018
 #9 [ffff88008b0f7f10] do_exit at ffffffff81052b11
#10 [ffff88008b0f7f40] do_group_exit at ffffffff81052c0d
#11 [ffff88008b0f7f70] sys_exit_group at ffffffff81052c92
#12 [ffff88008b0f7f80] system_call_fastpath at ffffffff81002ffb
    RIP: 00007f7c06d2c2a8  RSP: 00007fffd5e227a8  RFLAGS: 00010246
    RAX: 00000000000000e7  RBX: ffffffff81002ffb  RCX: 0000000000000000
    RDX: 0000000000000000  RSI: 000000000000003c  RDI: 0000000000000000
    RBP: 0000000000000000   R8: 00000000000000e7   R9: fffffffffffffa8
    R10: 00007f7c07212700  R11: 0000000000000246  R12: ffffffff81052c92
    R13: ffff88008b0f7f78  R14: 00007fffd5e237d4  R15: 0000000000000001
    ORIG_RAX: 00000000000000e7  CS: 0033  SS: 002b
```

观察RIP: ffffffff8108af68的回报指令，该指令用到(RAX+0x10)的地址。

```
crash> dis ffffffff8108af68
0xffffffff8108af68 <acct_collect+88>:   add    0x10(%rax),%rbx
```

观察这个地址的页表，可以发现PMD指向了0，说明这个地址的页表有问题。

```
crash> vtop ffff8800658ba6c8
VIRTUAL           PHYSICAL
ffff8800658ba6c8  658ba6c8

PML4 DIRECTORY: ffffffff81a04000
PAGE DIRECTORY: 1a05063
  PUD: 1a05008 => 100067
  PMD: 100960 => 0

      PAGE        PHYSICAL      MAPPING       INDEX CNT FLAGS
fffffea0001962e80  658ba000                0        0  1 100000000000100
```

由此，初步分析出页表被踩后，可以通过被非法访问内存空间的内容，推出可能的模块；或者通过crash时间点的附近日志分析可疑模块。

踩静态变量问题

分析方法：

通过“sym -l”命令可以看到内核符号，先找到被踩静态变量的地址，然后观察其附近地址还有哪些静态变量。常见的场景是其前面的某个变量，操作过程中溢出，造成周围变量都被踩。


```
ffffffff81a16100 (d) vm_table
ffffffff81a16d80 (d) fs_table
ffffffff81a17320 (d) debug_table
ffffffff81a173c0 (d) min_sched_granularity_ns
ffffffff81a173c4 (d) max_sched_granularity_ns
ffffffff81a173c8 (d) max_wakeup_granularity_ns
ffffffff81a173cc (d) min_sched_shares_ratelimit
ffffffff81a173d0 (d) max_sched_shares_ratelimit
ffffffff81a173d4 (d) max_sched_tunable_scaling
```

硬件 DMA 非法访问内存空间问题

1. 问题现象

   堆栈没有任何规律，唯一的相同点就是code区域是一样的，而且数值是有规律的，全部是“fe 0b ad ca”。

   ```
   [ 3051.054204] Call Trace:
   [ 3051.057035]  [<ffffffff8115faa9>] path_openat+0xd9/0x420
   [ 3051.063054]  [<ffffffff8115ff2c>] do_filp_open+0x4c/0xc0
   [ 3051.069103]  [<ffffffff81150b71>] do_sys_open+0x171/0x1f0
   [ 3051.075223]  [<ffffffff8144fc53>] ia32_do_call+0x13/0x13
   [ 3051.081261] Code: fe 0b ad ca fe 0b ad ca fe 0b ad ca fe 0b ad ca fe 0b ad ca fe 0b ad ca fe 0b ad ca fe 0b ad ca fe 0b ad ca fe 0b ad <ca> fe 0b ad ca fe 0b ad ca fe 0b ad ca fe 0b ad ca fe 0b ad ca
   ```

2. 分析方法：

   由于内核的code区是只读的，通过线性地址无法去修改，因此可以得出造成这个修改的必然使用的是物理地址，硬件DMA有这个可能。但是目标地址的内容与硬件设备的DMA没有直接联系，除了硬件DMA之外，BIOS有直接操作物理内存的能力，需要分析BIOS代码。

   最终发现驱动代码错误，DMA过程踩到BIOS，造成BIOS运行过程踩到内核内存。

死锁问题

1. 分析方法

   死锁问题，关键点是当时每个CPU的堆栈是什么。 通过“bt -a”命令可以打印出系统当时所有CPU运行的堆栈，通过解析锁的数据结构，可以分析出哪个线程持有锁。

2. 常见死锁类型：
   - spinlock保护区中执行了非原子性的流程，如sleep schedule流程。
   - spinlock保护区中执行逻辑消耗的时间太长。
   - AB-BA死锁。
   - AA死锁，重复上锁。
   - 环形锁，在某些复杂架构设计中，模块间的等待可能出现环形，这会造成死锁。
   - 上锁和解锁不对称，锁指针运行中会被修改。

3 监控告警使用指南

3.1 监控配置

3.1.1 简介

特性描述

监控框架负责监控OS系统运行过程中的异常，并将监控到的异常上报告警模块，并通过告警模块上报告警到产品的告警平台。监控框架以服务的形式提供，可以通过systemctl start|stop|restart|reload sysmonitor启动、关闭、重启、重载服务。

注意事项

- 监控框架不支持并发执行。
- 各配置文件须合法配置，否则可能造成监控框架异常。

配置总览

监控框架有一个主配置文件（/etc/sysconfig/sysmonitor），用于配置各监控项的监控周期、是否需要监控及异常时是否需要上报告警等。

说明

配置项的=和"之间不能有空格。

```
# 关键进程监控开关
PROCESS_MONITOR="on"

# 关键进程监控周期（秒）
PROCESS_MONITOR_PERIOD="3"

# 关键进程恢复失败后再次尝试拉起周期（分）
```



```
PROCESS_RECALL_PERIOD="1"

# 关键进程服务异常恢复过程中超时时间（秒）
PROCESS_RESTART_TIMEOUT="90"

# 关键进程告警开关
PROCESS_ALARM="off"

# 文件系统故障监控开关
FILESYSTEM_MONITOR="on"

# 文件系统故障告警开关
FILESYSTEM_ALARM="off"

# 信号监控开关
SIGNAL_MONITOR="on"

# 信号告警开关
SIGNAL_ALARM="off"

# 磁盘空间监控开关
DISK_MONITOR="on"

# 磁盘空间告警开关
DISK_ALARM="off"

# 磁盘监控周期（秒）
DISK_MONITOR_PERIOD="60"

# 磁盘inode监控开关
INODE_MONITOR="on"

# 磁盘inode告警开关
INODE_ALARM="off"

# 磁盘inode监控周期（秒）
INODE_MONITOR_PERIOD="60"

# 网卡监控开关
NETCARD_MONITOR="on"

# 网卡告警开关
NETCARD_ALARM="off"

# 文件监控开关
FILE_MONITOR="on"

# 文件告警开关
FILE_ALARM="off"

# CPU监控开关
CPU_MONITOR="on"

# CPU告警开关
CPU_ALARM="off"

# 内存监控开关
MEM_MONITOR="on"

# 内存告警开关
MEM_ALARM="off"

# 进程（线程）数监控开关
PSCNT_MONITOR="on"

# 进程（线程）数告警开关
PSCNT_ALARM="off"

# 系统fd总数监控开关
```

```
FDCNT_MONITOR="on"

# 系统fd总数告警开关
FDCNT_ALARM="off"

# DAEMON类型自定义监控开关
CUSTOM_DAEMON_MONITOR="on"

# periodic类型自定义监控开关
CUSTOM_PERIODIC_MONITOR="on"

# 本地磁盘IO延时监控开关
IO_DELAY_MONITOR="off"

# 本地磁盘IO延时告警开关
IO_DELAY_ALARM="off"

# 单个进程句柄数监控开关
PROCESS_FD_NUM_MONITOR="on"

# 单个进程句柄数告警开关
PROCESS_FD_NUM_ALARM="off"

# 重新拉起sysalarm次数（次）
# restart sysalarm times (range: 1-1000, unit: times, default: 3)
RESTART_ALARM_TIMES="3"

# 重新拉起sysalarm周期（秒）
# restart sysalarm period (range: 1-60, unit: sec, default: 3)
RESTART_ALARM_PERIOD="3"
```

各配置项说明见表3-1。

表 3-1 配置说明

配置项	配置项说明	是否必配	默认值
PROCESS_MONITOR	设定是否开启关键进程监控，on为开启，off为关闭	否	开启
PROCESS_MONITOR_PERIOD	设定关键进程监控的周期，单位秒	否	3s
PROCESS_RECALL_PERIOD	关键进程恢复失败后再次尝试拉起周期，单位分，取值范围为1至1440之间的整数	否	1min
PROCESS_RESTART_TIMEOUT	关键进程服务异常恢复过程中超时时间，单位秒，取值范围为30至300之间的整数 **说明** 例如因DNS配置的ip不通等原因导致rsyslog进程恢复时间较长时，需调整该配置项的值	否	90s

配置项	配置项说明	是否必配	默认值
PROCESS_ALARM	设定是否开启关键进程异常告警，on为开启，off为关闭	否	关闭
FILESYSTEM_MONITOR	设定是否开启ext3/ext4文件系统监控，on为开启，off为关闭	否	开启
FILESYSTEM_ALARM	设定是否开启ext3/ext4文件系统异常告警，on为开启，off为关闭	否	关闭
SIGNAL_MONITOR	设定是否开启信号监控，on为开启，off为关闭	否	开启
SIGNAL_ALARM	设定是否开启信号监控异常告警，on为开启，off为关闭	否	关闭
DISK_MONITOR	设定是否开启磁盘空间监控，on为开启，off为关闭	否	开启
DISK_ALARM	设定是否开启磁盘空间告警，on为开启，off为关闭	否	关闭
DISK_MONITOR_PERIOD	设定磁盘监控周期，单位秒	否	60s
INODE_MONITOR	设定是否开启磁盘inode监控，on为开启，off为关闭	否	开启
INODE_ALARM	设定是否开启磁盘inode告警，on为开启，off为关闭	否	关闭
INODE_MONITOR_PERIOD	设定磁盘inode监控周期，单位秒	否	60s
NETCARD_MONITOR	设定是否开启网卡监控，on为开启，off为关闭	否	开启
NETCARD_ALARM	设定是否开启网卡监控异常告警，on为开启，off为关闭	否	关闭

配置项	配置项说明	是否必配	默认值
FILE_MONITOR	设定是否开启文件监控，on为开启，off为关闭	否	开启
FILE_ALARM	设定是否开启文件监控异常告警，on为开启，off为关闭	否	关闭
CPU_MONITOR	设定是否cpu监控，on为开启，off为关闭	否	开启
CPU_ALARM	设定是否cpu告警，on为开启，off为关闭	否	关闭
MEM_MONITOR	设定是否内存监控，on为开启，off为关闭	否	开启
MEM_ALARM	设定是否内存告警，on为开启，off为关闭	否	关闭
PSCNT_MONITOR	设定是否进程数监控，on为开启，off为关闭	否	开启
PSCNT_ALARM	设定是否进程数告警，on为开启，off为关闭	否	关闭
FDCNT_MONITOR	设定是否fd总数监控，on为开启，off为关闭	否	开启
FDCNT_ALARM	设定是否fd总数告警，on为开启，off为关闭	否	关闭
CUSTOM_DAEMON_MONITOR	用户自定义的daemon类型的监控项，on为开启，off为关闭	否	开启
CUSTOM_PERIODIC_MONITOR	用户自定义的periodic类型的监控项，on为开启，off为关闭	否	开启
IO_DELAY_MONITOR	本地磁盘IO延时监控开关，on为开启，off为关闭	否	关闭

配置项	配置项说明	是否必配	默认值
IO_DELAY_ALARM	本地磁盘IO延时告警开关，on为开启，off为关闭	否	关闭
PROCESS_FD_NUM_MONITOR	设定是否开启单个进程句柄数监控，on为开启，off为关闭	否	开启
PROCESS_FD_NUM_ALARM	设定是否开启单个进程句柄数告警，on为开启，off为关闭	否	关闭
RESTART_ALARM_TIMES	sysalarm异常，重新拉起sysalarm的次数 **说明** ● 如果在配置的范围内sysalarm还未被拉起，则不会再重拉。 ● 拉起成功后如果sysalarm再出现异常，则重新计数。 ● 日志只记录前5次，或10的倍数次	否	3次 **说明** 拉起次数范围1-1000次，默认3次
RESTART_ALARM_PERIOD	sysalarm异常，重新拉起sysalarm的周期	否	3s **说明** 拉起周期范围1-60s，默认3秒

注意

1. 修改/etc/sysconfig/sysmonitor配置文件后，需要重启sysmonitor服务生效。
2. 配置文件中，如果某一项没有配置，默认为监控项开启，告警项关闭。

目前版本中支持ext3/ext4文件系统故障、关键进程模块监控、磁盘空间监控、磁盘inode监控、文件监控、信号监控、网卡监控、cpu占用率监控、内存监控、进程数监控、用户自定义监控。

表 3-2 监控项列表

监控项	是否可配
ext3/ext4文件系统故障	否

监控项	是否可配
关键进程/模块异常	是
磁盘空间	是
磁盘inode	是
文件	是
信号	是
网卡	是
cpu占用率	是
内存	是
进程数	是
自定义监控	是

命令参考

- 启动监控服务：
  ```
  systemctl start sysmonitor
  ```
- 关闭监控服务：
  ```
  systemctl stop sysmonitor
  ```
- 重启监控服务：
  ```
  systemctl restart sysmonitor
  ```
  说明

  修改/etc/sysconfig/sysmonitor后，须重启服务修改后的配置才能生效。
- 修改监控项的配置文件后，重载监控服务可使修改后的配置动态生效：
  ```
  systemctl reload sysmonitor
  ```
  说明

  重载操作对网卡监控不生效，修改完网卡监控的配置文件须重启服务才能生效。

3.1.2 ext3/ext4 文件系统监控

简介

当文件系统出现故障时会导致IO操作异常从而引发操作系统一系列问题。通过文件系统故障检测及时发现文件系统故障，并上报告警给产品，以便系统管理员或用户及时恢复故障。

配置文件说明

无。

异常日志

对于增加了errors=remount-ro挂载选项的文件系统，如果监控到ext3/ext4文件系统故障，sysmonitor.log中打印异常信息示例如下：


```
[2015-03-16:18:41:38]sysmonitor[5026]: sdc1 filesystem error. Remount filesystem read-only.
```

其他异常场景下，如果监控到ext3/ext4文件系统故障，sysmonitor.log中打印异常信息示例如下：

```
[2015-03-16:18:41:38]sysmonitor[6172]: sda6 filesystem error.
```

3.1.3 关键进程监控

简介

定期监控系统中的关键进程，当系统内关键进程异常退出时，自动尝试恢复关键进程。如果恢复失败并需要告警，可上报告警。系统管理员能及时感知进程异常退出事件，以及进程是否被恢复拉起。问题定位人员能从日志中定位进程异常退出的时间。

配置文件说明

配置目录为/etc/sysmonitor/process，每个进程或模块一个配置文件。

配置文件示例如下：

```
NAME=crond
RECOVER_COMMAND=systemctl restart crond
MONITOR_COMMAND=systemctl status crond
STOP_COMMAND=systemctl stop crond
```

各配置项说明见表3-3。

表 3-3 配置项说明

配置项	配置项说明	是否必配	默认值
NAME	进程或模块名	是	无
RECOVER_COMMAND	恢复命令	是	无
MONITOR_COMMAND	监控命令 **说明** 命令返回值为0视为进程正常，命令返回大于0视为进程异常。	否	pgrep -f $(which xxx) **说明** “xxx”为NAME字段中配置的进程名。
STOP_COMMAND	停止命令 **说明** 建议添加，不然存在子进程残留的风险。	否	无

注意

1. 修改关键进程监控的配置文件后，须执行systemctl reload sysmonitor，新的配置在一个监控周期后生效。
2. 恢复命令和监控命令不能阻塞，否则造成关键进程监控线程异常。
3. 当恢复命令执行超时90s时，会调用停止命令终止进程。
4. 当关键进程异常后，并且尝试拉起三次都不成功，最终会按照全局配置文件中配置的PROCESS_RECALL_PERIOD周期进行拉起。
5. 若监控的进程不是daemon进程，MONITOR_COMMAND必配。
6. 若配置的关键服务在当前系统上不存在或者配置错误，则该监控不会生效，日志中会有相应提示。
7. 配置文件的权限建议为600，监控项建议为systemd中的service类型（如MONITOR_COMMAND=systemctl status crond），若监控的为进程，请确保NAME字段为绝对路径。
8. sysmonitor重启（restart）、重载（reload）、退出（stop）都不会影响所监控的进程或服务。

异常日志

如果监控到进程或模块异常，/var/log/sysmonitor.log中打印异常信息示例如下：

```
[2015-03-16:18:41:38]sysmonitor[5018]: crond is abornmal, check cmd return 1, use "systemctl restart crond" to recover
[2015-03-16:18:41:38]sysmonitor[5018]: crond is abnormal, check cmd return 1
```

3.1.4 文件监控

简介

系统关键文件被意外删除后，会导致系统运行异常甚至崩溃。通过文件监控可以及时获知系统中关键文件被删除或者有恶意文件被添加，并且上报告警给产品，以便管理员和用户及时获知并处理故障。

配置文件说明

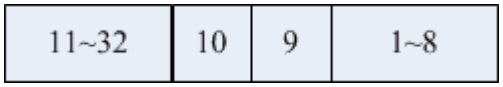
配置文件为/etc/sysmonitor/file。每个监控配置项为一行，监控配置项包含两个内容：监控文件（目录）和监控事件。监控文件（目录）是绝对路径，监控文件（目录）和监控事件中间由一个或多个空格隔开。

注意

1. 由于告警信息和日志长度限制，建议监控的文件或目录绝对路径长度小于等于223。如果配置的监控对象绝对路径长度超过223，可能会有日志打印不完整的现象出现。
2. 请用户自行确保监控文件路径正确，如果配置文件不存在或错误则无法监控到该文件。
3. 由于系统路径长度限制，监控的文件或目录绝对路径长度必须小于4096。
4. 支持监控目录和常规文件，/proc和/proc/*、/dev和/dev/*、/sys和/sys/*、管道文件、socket文件等均不支持监控。
5. /var/log和/var/log/*均只支持删除事件。
6. 当配置文件中存在多条相同路径的时候，以第一条合法配置为准，其他相同配置均不生效。在日志文件中可以查看到其他相同配置被忽略的提示。
7. 不支持对软链接配置监控；当配置硬链接文件的删除事件时，需删除该文件和它的全部硬链接才会打印文件删除事件。
8. 当文件添加监控成功及监控的事件发生时，监控日志打印的是配置文件中路径的绝对路径。

监控文件（目录）采用了位图的方式配置要监控的事件，对文件或目录进行监控的事件位图如图3-1所示。

图 3-1 监控事件位图



事件位图中每一位代表一个事件。第n位如果置1，则表示监控第n位对应的事件；第n位如果置0，则表示不监控第n位对应的事件。监控位图对应的16进制数，即是写到配置文件中的监控事件项。目前每一位对应的事件见表3-4。

表 3-4 文件事件位图中的位与事件对应关系表

配置项	配置项说明	是否必配
1～8	保留	否
9	文件、目录添加事件	是
10	文件、目录删除事件	是
11～32	保留	否

注意

1. 修改文件监控的配置文件后，须执行systemctl reload sysmonitor，新的配置在最多60秒后生效。
2. 监控事件需要严格遵守上述规则，如果配置有误，则无法监控；如果配置项中监控事件填空，则默认只监控删除事件，即0x200。
3. 文件或目录删除后，只有当所有打开该文件的进程都停止后才会上报删除事件。
4. 监控的文件通过vi、sed等操作修改后会在监控日志中打印File "XXX" may have been changed。

文件监控目前实现了对添加和删除事件的监控，即第9和第10位有效，其他位为保留位，暂不生效。如果配置事件配置了1-8、11-32位为1，监控日志中会提示监控事件配置错误。

示例

配置对/home下子目录的增加和删除事件监控，低12位位图为：001100000000，则可以配置如下：

```
/home      0x300
```

配置对/etc/ssh/sshd_config文件的删除事件监控，低12位位图为：001000000000，则可以配置如下：

```
/etc/ssh/sshd_config  0x200
```

异常日志

如果监控文件有配置的事件发生，/var/log/sysmonitor.log中打印日志示例如下：

```
[2015-03-16:18:41:38]sysmonitor[3373]: 1 events queued
[2015-03-16:18:41:38]sysmonitor[3373]: 1th event handled
[2015-03-16:18:41:38]sysmonitor[3373]: Subdir "b" under "/home/a" was added.
[2015-03-16:18:41:38]sysmonitor[3373]: 1 events queued
[2015-03-16:18:41:38]sysmonitor[3373]: 1th event handled
[2015-03-16:18:41:38]sysmonitor[3373]: Subfile "c" under "/home/a" was deleted.
```

3.1.5 信号监控

简介

当进程被SIGKILL或SIGTERM信号异常终止时，通过记录信号发送的进程以及发送的信号以便于问题定位。

配置文件说明

配置文件为/etc/sysmonitor/signal，示例如下：

```
SIGKILL="on"

SIGTERM="on"
```

各配置项说明见表3-5。

表 3-5 配置项说明

配置项	配置项说明	是否必配	默认值
SIGKILL	SIGKILL信号	否	on
SIGTERM	SIGTERM信号	否	on
SIGHUP	SIGHUP信号	否	off
SIGINT	SIGINT信号	否	off
SIGQUIT	SIGQUIT信号	否	off
SIGILL	SIGILL信号	否	off
SIGTRAP	SIGTRAP信号	否	off
SIGABRT	SIGABRT信号	否	off
SIGBUS	SIGBUS信号	否	off
SIGFPE	SIGFPE信号	否	off
SIGUSR1	SIGUSR1信号	否	off
SIGSEGV	SIGSEGV信号	否	off
SIGUSR2	SIGUSR2信号	否	off
SIGPIPE	SIGPIPE信号	否	off
SIGALRM	SIGALRM信号	否	off
SIGSTKFLT	SIGSTKFLT信号	否	off
SIGCHLD	SIGCHLD信号	否	off
SIGCONT	SIGCONT信号	否	off
SIGSTOP	SIGSTOP信号	否	off
SIGTSTP	SIGTSTP信号	否	off
SIGTTIN	SIGTTIN信号	否	off
SIGTTOU	SIGTTOU信号	否	off
SIGURG	SIGURG信号	否	off
SIGXCPU	SIGXCPU信号	否	off
SIGXFSZ	SIGXFSZ信号	否	off
SIGVTALRM	SIGVTALRM信号	否	off
SIGPROF	SIGPROF信号	否	off
SIGWINCH	SIGWINCH信号	否	off
SIGIO	SIGIO信号	否	off

配置项	配置项说明	是否必配	默认值
SIGPWR	SIGPWR信号	否	off
SIGSYS	SIGSYS信号	否	off

注意

修改信号监控的配置文件后，须执行systemctl reload sysmonitor后生效。

异常日志

如果监控到配置的信号事件，/var/log/sysmonitor.log中打印信息示例如下：

```
[2015-03-16:18:41:38]sysmonitor[7965]: sshd[18883](parent:sshd[944]) send SIGKILL to sshd[18884]
```

3.1.6 磁盘分区监控

简介

定期监控系统中挂载的磁盘分区空间，当磁盘分区使用率大于或等于用户设置的告警阀值，上报磁盘空间告警。发生告警后，当磁盘分区使用率小于用户设置的告警恢复阀值，上报磁盘空间恢复告警。

配置文件说明

配置文件为/etc/sysmonitor/disk。

配置文件示例如下：

```
DISK="/var/log" ALARM="90" RESUME="80"
DISK="/" ALARM="95" RESUME="85"
```

各配置项说明见表3-6。

表 3-6 配置项说明

配置项	配置项说明	是否必配	默认值
DISK	磁盘挂载目录名 **说明** 必须为磁盘挂载点或被挂载的磁盘分区。 最大长度为64字节。	是	无

配置项	配置项说明	是否必配	默认值
ALARM	整数，磁盘空间告警阈值 **说明** 0-100的整数，不要有空格等其他额外字符。	否	90
RESUME	整数，磁盘空间恢复阈值 **说明** 0-100的整数，不要有空格等其他额外字符。	否	80

注意

1. 修改磁盘空间监控的配置文件后，须执行systemctl reload sysmonitor，新的配置在一个监控周期后生效。
2. 重复配置的挂载目录，最后一个配置项生效。
3. ALARM值应该大于RESUME值。
4. 只能针对挂载点或被挂载的磁盘分区做监控。
5. 在CPU和I/O高压场景下，df执行命令超时，会导致磁盘利用率获取不到。
6. 当多个挂载点对应同一个磁盘分区时，以挂载点为准来上报告警

异常日志

如果监控到磁盘空间告警，/var/log/sysmonitor.log中打印信息示例如下：

```
[2015-03-16:18:41:38]sysmonitor[5018]: report disk alarm, /var/log used:90% alarm:90%
[2015-03-16:18:41:38]sysmonitor[5018]: report disk recovered, /var/log used:79% resume:80%
```

3.1.7 网卡状态监控

简介

系统运行过程中可能出现因人为原因或异常而导致网卡状态或IP发生改变，对网卡状态和IP变化进行监控，以便及时感知到异常并方便定位异常原因。

配置文件说明

配置文件为/etc/sysmonitor/network，示例如下：

```
#dev event
eth1 UP
```

各配置项说明见表3-7。

表 3-7 配置项说明

配置项	配置项说明	是否必配	默认值
dev	网卡名	是	无
event	侦听事件，可取值为UP、DOWN、NEWADDR、DELADDR。 ● UP：网卡UP ● DOWN：网卡DOWN ● NEWADDR：增加IP地址 ● DELADDR：删除IP地址	否	若侦听事件为空则UP、DOWN、NEWADDR、DELADDR都监控。

注意

1. 修改网卡监控的配置文件后，须执行systemctl restart sysmonitor，新的配置才可生效。
2. 如果配置文件为空，则不监控网卡事件；如果只配置网卡名称不配置侦听事件，则所有事件都监控。
3. 目前仅支持监控IPv4地址的添加和删除事件。
4. 不支持虚拟网卡UP和DOWN的状态监控。
5. 请确保网卡监控的配置文件每行少于197个字符，若超过197个字符会在监控日志中打印配置错误的提示信息。

异常日志

如果监控到配置的网卡事件，/var/log/sysmonitor.log中打印信息示例如下：

```
[2015-03-16:18:41:38]sysmonitor[5018]: eth1: device is up
[2015-03-16:18:41:38]sysmonitor[5018]: eth1: ip[192.168.0.1] prefixlen[24] is added
```

3.1.8 cpu 监控

简介

监控系统cpu占用情况，当cpu使用率超出或低于阈值时，记录日志或上报告警。

配置文件说明

配置文件为/etc/sysmonitor/cpu。

配置文件示例如下：

```
# 告警产生百分比上限
ALARM="90"
```

```
# 告警恢复百分比下限
RESUME="80"

# 监控周期（秒）
MONITOR_PERIOD="60"

# 统计周期（秒）
STAT_PERIOD="300"
```

表 3-8 配置项说明

配置项	配置项说明	是否必配	默认值
ALARM	大于0，cpu占用率告警阈值。	否	90
RESUME	大于等于0，cpu占用率恢复阈值。	否	80
MONITOR_PERIOD	监控周期（秒），取值大于0。	否	60
STAT_PERIOD	统计周期（秒），取值大于0。	否	300

注意

1. 修改cpu监控的配置文件后，须执行systemctl reload sysmonitor，新的配置在一个监控周期后生效。
2. ALARM值应该大于RESUME值。

异常日志

如果监控到cpu告警，/var/log/sysmonitor.log中打印信息示例如下：

```
[2015-04-16:06:18:54]sysmonitor[27482]: CPU usage alarm:  91.3%
[2015-04-16:06:24:16]sysmonitor[31896]: CPU usage resume:  70.1%
```

3.1.9 内存监控

简介

监控系统内存占用情况，当内存使用率超出或低于阈值时，记录日志或上报告警。

配置文件说明

配置文件为/etc/sysmonitor/memory。

配置文件示例如下：

```
# 告警产生百分比上限
ALARM="90"

# 告警恢复百分比下限
```

```
RESUME="80"

# 监控周期（秒）
PERIOD="60"
```

表 3-9 配置项说明

配置项	配置项说明	是否必配	默认值
ALARM	大于0，内存占用率告警阈值。	否	90
RESUME	大于等于0，内存占用率恢复阈值。	否	80
PERIOD	监控周期（秒），取值大于0。	否	60

注意

1. 修改内存监控的配置文件后，须执行systemctl reload sysmonitor，新的配置在一个监控周期后生效。
2. ALARM值应该大于RESUME值。
3. 取三个监控周期的内存占用的平均值，来作为是否上报发生告警或者恢复告警的依据。

异常日志

如果监控到内存告警，/var/log/sysmonitor.log中打印信息示例如下：

```
[2015-04-16:07:24:23]sysmonitor[31896]: memory usage alarm:  90.1%
[2015-04-16:07:28:16]sysmonitor[31896]: memory usage resume:  60.4%
```

3.1.10 进程数监控

简介

监控系统进程数目，当进程总数超出或低于阈值时，记录日志或上报告警。

配置文件说明

配置文件为/etc/sysmonitor/pscnt。

配置文件示例如下：

```
# 进程（包括线程）数告警产生上限(取该值和pid_max的1%中的最大值)
ALARM="1600"

# 进程（包括线程）数告警恢复下限(取该值和pid_max的0.95%中的最大值)
RESUME="1500"

# 监控周期（秒）
PERIOD="60"
```


表 3-10 配置项说明

配置项	配置项说明	是否必配	默认值
ALARM	大于0的整数，进程总数告警阈值。	否	1600
RESUME	大于等于0的整数，进程总数恢复阈值。	否	1500
PERIOD	监控周期（秒），取值大于0。	否	60

注意

1. 修改进程数监控的配置文件后，须执行systemctl reload sysmonitor，新的配置在一个监控周期后生效。
2. ALARM值应该大于RESUME值。
3. 进程数告警产生阀值取ALARM值与/proc/sys/kernel/pid_max的1%中的最大值，告警恢复阀值取RESUME值与/proc/sys/kernel/pid_max的0.95%中的最大值。

异常日志

如果监控到进程数告警，/var/log/sysmonitor.log中打印信息示例如下：

```
[2015-04-16:07:44:54]sysmonitor[31896]: process count alarm: 1657
[2015-04-16:07:45:17]sysmonitor[31896]: process count resume: 1200
```

3.1.11 系统句柄总数监控

简介

监控系统文件句柄（fd）数目，当系统文件句柄总数超出或低于阈值时，记录日志并上报告警或恢复告警，同时记录当前系统句柄峰值到sysmonitor.log

1、当前系统文件句柄总数超过阈值后，上报告警，并在监控日志中打印，当前系统所有句柄数

2、当前系统文件句柄总数低于恢复告警阈值后，恢复告警，并打印当前系统所有句柄数。

配置文件说明

配置文件为/etc/sysmonitor/sys_fd_conf

配置文件示例如下：

```
# 系统fd总数百分比上限
SYS_FD_ALARM="80"

# 系统fd总数百分比下限
SYS_FD_RESUME="70"
```



```
# 监控周期（100~86400秒）
SYS_FD_PERIOD="600"
```

表 3-11 配置项说明

配置项	配置项说明	是否必配	默认值
SYS_FD_ALARM	大于0小于100的整数，fd总数与系统最大fd数百分比的告警阈值。	否	80%
SYS_FD_RESUME	大于0小于100的整数，fd总数与系统最大fd数百分比的恢复阈值。	否	70%
SYS_FD_PERIOD	监控周期（秒），取值为100~86400之间的整数。	否	600

注意

1. 修改fd总数监控的配置文件后，须执行systemctl reload sysmonitor，新的配置在一个监控周期后生效。
2. SYS_FD_ALARM值应该大于SYS_FD_RESUME值，当配置非法时，会使用默认值，并打印日志。
3. 系统复位开关：/proc/files_panic_enbale为1（默认值为0，关闭状态）时，系统句柄数耗尽会复位系统。用户可通过以下命令打开开关：

```
echo 1 > /proc/files_panic_enbale
```

异常日志

如果监控到fd总数告警，在监控日志中打印告警。sysmonitor.log中打印信息示例如下：

```
[2018-04-27:23:58:06]sysmonitor[14340]: sys fd count alarm: 259296
```

3.1.12 单个进程句柄数监控

简介

监控系统单个进程句柄数目，当单个进程句柄总数超出阈值时，记录日志或上报事件，并且记录峰值到单独的文件。

配置文件说明

配置文件为/etc/sysmonitor/process_fd_num。

配置文件示例如下：

```
#单个进程句柄告警阀值 单位%
PR_FD_ALARM="80"
```

表 3-12 配置项说明

配置项	配置项说明	是否必配	默认值
PR_FD_ALARM	大于0小于100的整数，单个进程句柄数与系统单个进程最大句柄数百分比的事件上报阈值。	否	80%

说明

1. 修改单个进程句柄监控的配置文件后，须执行systemctl reload sysmonitor。
2. 单个进程监控到的进程名，最多显示16个字节。

异常日志

如果监控到单个进程句柄事件发生，会在监控日志中记录。sysmonitor.log中打印信息示例如下：

```
[2018-04-27:23:52:18]sysmonitor[14340]: pid [28738] cmd [a.out]  fd more than [717]
```

3.1.13 磁盘 inode 监控

简介

定期监控系统中挂载的磁盘分区的inode，当磁盘分区的inode使用率大于或等于用户设置的告警阀值，上报磁盘inode告警。发生告警后，当磁盘分区inode使用率小于用户设置的告警恢复阀值，上报磁盘inode恢复告警。

配置文件说明

配置文件为/etc/sysmonitor/inode。

配置文件示例如下：

```
DISK="/var/log" ALARM="90" RESUME="80"
DISK="/" ALARM="95" RESUME="85"
```

各配置项说明见表3-13。

表 3-13 配置项说明

配置项	配置项说明	是否必配	默认值
DISK	磁盘挂载目录名 **说明** 必须为磁盘挂载点或被挂载的磁盘分区。 最大长度为64字节。	是	无

配置项	配置项说明	是否必配	默认值
ALARM	整数，磁盘inode告警阈值 **说明** 0-100的整数，不要有空格等其他额外字符。	否	90
RESUME	整数，磁盘inode恢复阈值 **说明** 0-100的整数，不要有空格等其他额外字符。	否	80

注意

1. 修改磁盘inode监控的配置文件后，须执行systemctl reload sysmonitor，新的配置在一个监控周期后生效。
2. 重复配置的挂载目录，最后一个配置项生效。
3. ALARM值应该大于RESUME值。
4. 只能针对挂载点或被挂载的磁盘分区做监控。
5. 在CPU和I/O高压场景下，df执行命令超时，会导致磁盘inode利用率获取不到。
6. 当多个挂载点对应同一个磁盘分区时，以挂载点为准来上报告警。

异常日志

如果监控到磁盘inode告警，/var/log/sysmonitor.log中打印信息示例如下：

```
[2015-03-16:18:41:38]sysmonitor[5018]: report disk inode alarm, /var/log used:90% alarm:90%
[2015-03-16:18:41:38]sysmonitor[5018]: report disk inode recovered, /var/log used:79% resume:80%
```

3.1.14 本地磁盘 IO 延时监控

简介

每5秒读取一次本地磁盘IO延时数据，每五分钟对在该5分钟内60组数据进行统计，如果有多于30次（一半）的数据大于配置的最大IO延时数据，则上报该磁盘的IO延时过大告警。

配置文件说明

配置文件为/etc/sysmonitor/iodelay。

配置文件示例如下：

```
DELAY_VALUE="100"
```

各配置项说明见表3-14。

表 3-14 配置项说明

配置项	配置项说明	是否必配	默认值
DELAY_VALUE	磁盘IO延时的最大值 **说明** 默认为100ms，配置的值默认单位是毫秒。 最大长度为64字节。 所有磁盘都以该值为最大值	是	100

注意

1. 修改本地磁盘io延时监控的配置文件后，须执行重载sysmonitor命令，新的配置在一个监控周期后生效。
2. 默认告警为最近60次延时平均大小有30次大于设置的阀值，恢复告警为最近60次延时平均大小有30次小于等于设置的阀值。

异常日志

如果监控到本地磁盘IO延时过大告警和告警恢复，/var/log/sysmonitor.log中打印信息示例如下：

```
[2016-10-16:18:41:38]sysmonitor[5018]: local disk: sda IO delay is too large
[2015-03-16:18:46:38]sysmonitor[5018]: local disk: sda IO delay is normal
```

3.1.15 自定义监控

简介

用户可以自定义监控项，监控框架读取配置文件内容，解析配置文件各监控属性，在监控框架里调用用户要执行的监控动作。监控模块仅提供监控框架，不感知用户要监控的内容以及如何监控，不负责告警上报。

配置文件说明

配置目录为/etc/sysmonitor.d/，每个进程或模块一个配置文件。

配置文件示例如下：

```
MONITOR_SWITCH="on"
TYPE="daemon"
EXECSTART="/etc/sysmonitor.d/uvpmonitor/unetwork_alarm.py"
ENVIROMENTFILE="/etc/sysmonitor.d/uvpEnvironmentFiles/unetwork-alarm.conf"
MONITOR_SWITCH="on"
TYPE="periodic"
EXECSTART="python /etc/sysmonitor.d/uvpmonitor/ipmi_monitor.py"
PERIOD="30"
```

各配置项说明见表3-15。

表 3-15 配置项说明

配置项	配置项说明	是否必配	默认值
MONITOR_SWITCH	监控开关	否	off
TYPE	自定义监控项的类型	是	无
EXECSTART	执行监控命令	是	无
ENVIROMENTFILE	环境变量存放文件	否	无
PERIOD	若type为periodic类型，此为必配项，为自定义监控的周期，取值为大于0的整数。	periodic类型为必配项	无

注意

1. 配置文件名称，环境变量文件名称，加上绝对路径总长度不能超过127个字符。
2. EXECSTART项的命令总长度不能超过159个字符，关键字段配置不能有空格。
3. 周期性监控的执行命令不能超时，否则对自定义监控框架产生影响。
4. 目前支持配置的环境变量最多为64个。
5. daemon类型的自定义监控每间隔10s会统一查询是否有reload命令下发，或者是否有daemon进程异常退出；如果有reload命令下发，需要等待10s后才会重新加载新的配置，如果有daemon进程异常退出，需要等待10s后才会重新拉起。
6. 关键字ENVIROMENTFILE对应的文件中的内容发生变化，如新增环境变量，或环境变量的值发生改变，需要重启sysmonitor服务，新的环境变量才能生效。
7. /etc/sysmonitor.d/目录下的配置文件权限建议为600，EXECSTART项中若只配置了执行文件，则执行文件的权限建议为550。

异常日志

如果daemon类型监控项异常退出，会记录log：

```
[2016-09-07:01:25:33]sysmonitor[22520]: custom daemon monitor: child process[11609] name
unetwork_alarm exit code[127], [1] times.
```

3.2 告警上报

3.2.1 如何上报告警

简介

alarm模块用来提供函数接口用于上报告警到sysalarm，除了支持sysmonitor服务通过调用该模块上报告警到sysalarm外，其他用户也可以将自定义的告警信息上报到

sysalarm，被上报的告警信息最后统一由注册告警模块接收并处理（注册告警部分见3.2.2 如何注册接收告警）。现支持C和shell接口上报告警。告警模块对于上报的重复的告警具有抑制功能，具体抑制时间用户可以配置。告警模块会保存已经上报告警的最新状态（恢复或产生告警），如果sysalarm发生重启，这些告警信息可以恢复。

告警消息的组成

- 上报告警的消息组成

告警项	数据类型	描述
usAlarmId	unsigned short	告警id，某一类故障一个id
ucAlarmLevel	unsigned char	告警级别，表示：(1)紧急、(2)重要、(3)次要、(4)提示、(5)不确定
ucAlarmType	unsigned char	告警类别，表示：(0)告警恢复、(1)告警产生、(2)事件发生
pucParas	unsigned char*	告警描述信息
pucExParas	unsigned char*	告警拓展描述信息

- 注册告警接口定义

C语言接口定义

```
int os_alarm(unsigned short usAlarmId, unsigned char ucAlarmLevel, unsigned char ucAlarmType,
unsigned char* pucParas, unsigned char* pucExparas);
```

参数详细说明：

- usAlarmId：告警Id，取值范围为[1001, 2000]。
- ucAlarmLevel：告警级别，用数字1到5表示，意义如下：
  1： Critical（紧急）
  2： Major（重要）
  3： Minor（次要）
  4： Warning（提示）
  5： Indeterminate（不确定）
- ucAlarmType：告警类型，数字0，1，2表示，意义如下：
  0： 告警恢复
  1： 告警产生
  2： 事件
- pucParas：描述信息，字符串，最大长度为255。
- pucExParas：拓展信息，字符串，最大长度为255。

代码示例如下：

```
#include <stdio.h>
int main()
{
    unsigned short alarmid = 1100; //告警id
    unsigned char alarmlevel = 3; //告警级别
```



```
    unsigned char alarmtype = 1; //告警类型
    unsigned char msg[] = "I am message"; //描述信息
    unsigned char exmsg[] = "I am ext message"; //拓展信息

    os_alarm(alarmid, alarmlevel, alarmtype, msg, exmsg);
    return 0;
}
```

shell接口定义

- sendalarm [-i alarmid] [-t alarmtype] [-l alarmlevel] [-m msg1] [-e msg2]
  - -i alarmid：alarmid为告警Id，取值范围为[1001, 2000]。
  - -t alarmtype：alarmtype为告警类型，只能填normal, abnormal或event。
  - -l alarmlevel：alarmlevel为告警级别，可填1,2,3,4,5。
  - -m msg1: msg1为描述信息，可填任意字符串，最大长度为255。
  - -e msg2: msg2为拓展信息，可填任意字符串，最大长度为255。

    代码示例如下：

    ```
    sendalarm -i 1100 -t abnormal -l 3 -m "I am message" -e "I am ext message"
    ```
- sendalarm ?

可用来查询sendalarm的用法。

```
linux-ErMkTw:/home # sendalarm ?
usage:  sendalarm [-i id] [-t type] [-l level] [-m message] [-e extend message]
        -i  alarm id
        -t  alarm type, could be: normal, abnormal, event
        -l  alarm level, could be 1(critial),2(major),3(minor),4(warning) or 5(indeterminate)
        -m  alarm message
        -e  alarm extend message
```

注意事项

- 告警Id取值范围有限制，为[1001,2000]，超出此范围的告警ID，认为是没有注册的非法告警，不进行告警上报处理。
- 上报接口在libalarm.so动态库中实现，请确保系统中存在并能找到上述动态库。
- 接口的生效依赖告警转发守护进程sysalarm，使用时确保sysalarm进程已启动。
- 要使用alarm模块中的C语言接口，系统里必须包含libalarm.so文件。使用shell接口，必须包含sendalarm文件。

告警抑制

sysalarm对接收到的相同的告警信息，在设定的时间内具有抑制功能。抑制时间在/etc/sysconfig/sysalarm文件中进行配置，如下所示：

```
# 告警抑制时间（范围：0-30，单位：分钟，默认：15）
DEPRESSION_TIME="15"
```

在告警抑制的时间内出现相同的告警时，会被sysalarm抑制，告警不会再次上报。告警抑制时间的默认值为15（分钟），当用户配置的时间超出范围，或者输入非法时，系统中默认按照15来处理；若关键字DEPRESSION_TIME="XX"配置错误，或者配置文件为空，系统按照0来处理。如果不需要抑制功能，抑制时间可以配置成0。

判断是否为相同告警，主要通过三个告警信息来判断：告警ID，告警类型，告警扩展信息。

注意

对于事件类型的告警，如网卡监控告警、信号监控告警等，不支持告警抑制功能。

告警保存

sysalarm对于接收到的告警信息，能动态实时更新保存到内存文件中。同样的ID和告警关键信息只保存最后收到的一条告警的状态。最多保存的数量为10000条。

sysalarm服务重启后，能读取告警信息文件中保存的告警信息，恢复之前的告警状态。系统重启后，文件不存在则无需恢复。

提供shell命令，可以用于查看告警内存文件中的告警信息。命令为：transalarm，执行命令后，会生成一个可读的告警文件trans-alarm，存放在/var/log目录下。

告警删除

sysalarm支持告警删除，在某些场景下，告警对象已经不存在，需要删除保存在内存文件中的告警信息。用户可以调用删除接口，传入告警ID，以及时间，来删除对应的告警信息。sysalarm提供告警删除的C接口和shell接口。接口定义如下：

```
/*****************************************************************
Function     : os_alarm_del
Description  : 外部接口用于删除告警记录, 从当前时间算起，n分钟前的告警全部删除，n为AlarmTime, 单位为min，如果n为0，则将此ID的告警全部删除
Input        : AlarmId, AlarmTime
Output       : 无
Return       : 0:成功 -1:失败
*****************************************************************/
int os_alarm_del(unsigned short usAlarmId, int AlarmTime);
```

```
delalarm
usage: delalarm [-i id] [-t time]
       -i  alarm id
       -t  alarm time, eg. -t 4, it means alarm generated before 4 mins will be deleted
```

- 删除接口在libalarm.so动态库中实现，请确保系统中存在并能找到上述动态库。
- 接口的生效依赖告警转发守护进程sysalarm，使用时确保sysalarm进程已启动。
- 要使用alarm模块中的C语言接口，系统里必须包含libalarm.so文件。使用shell接口，必须包含delalarm文件

告警注册接收步骤

步骤1 安装支持监控告警特性的操作系统。

步骤2 检查系统中是否有libalarm.so文件，如果没有需要先下载安装libalarm包。

步骤3 检查sysalarm服务是否正常运行，若未运行，调用命令systemctl restart sysalarm 拉起sysalarm服务。

步骤4 调用接口函数（os_alarm 或 sendalarm）上报告警。

----结束

3.2.2 如何注册接收告警

简介

告警模块提供模块间的告警转发服务，对接收者提供告警回调函数的注册接口，用于传递告警信息到产品平台。当有告警信息上报后，告警模块回调注册函数接收告警信息并传递该信息到用户自定义的函数里进行处理。

告警注册接口依赖sysalarm服务，可以通过以下命令去启动、关闭、重启sysalarm服务：

```
systemctl start|stop|restart sysalarm
```

重拉功能

sysmonitor如果被异常终止，sysalarm能够重新拉起sysmonitor。配置文件/etc/sysconfig/sysalarm中，可配置sysalarm重拉sysmonitor的时间间隔RESTART_PERIOD和重拉总次数RESTART_TIMES，每两次重新拉起的时间间隔会随着重拉次数的增加而延长。

```
# restart sysmonitor period (range: 15-1000, unit: sec, default: 15)
RESTART_PERIOD="15"
# restart sysmonitor times (range: 3-1000, unit: times, default: 20)
RESTART_TIMES="20"
```

告警消息的组成

- 告警消息组成

告警项	数据类型	描述
usAlarmId	unsigned short	告警id，某一类故障一个id
ucAlarmLevel	unsigned char	告警级别，表示紧急、重要、次要、提示、不确定
ucAlarmType	unsigned char	告警类别，表示告警恢复、告警产生或事件发生
AlarmTime	struct timeval	接收告警时间
pucParas	unsigned char*	告警描述信息
pucExParas	unsigned char*	告警拓展描述信息

注册告警 C 语言接口

- 注册告警C语言接口定义

```
typedef int (*alarm_callback_func)(void *palarm);
typedef int (*alarm_report_func)(unsigned short, unsigned char, unsigned char, unsigned char*);
struct alarm_hook_struct
{
    alarm_report_func pfunc_alarmreport;
};
struct alarm_hook
{
alarm_callback_func pfunc_alarmcallbk;
};
int OS_alarm_Register(struct alarm_hook* alarm);
```



```
void OS_alarm_UnRegister(void);
unsigned short OS_alarm_getid(void *palarm);
unsigned char OS_alarm_gettype(void *palarm);
unsigned char OS_alarm_getlevel(void *palarm);
long long OS_alarm_gettime(void *palarm);
char * OS_alarm_getdesc(void *palarm);
char * OS_alarm_getexdesc(void *palarm);
```

注意事项

- C语言注册接口在libregalarm.so动态库中实现，libregalarm.so依赖libsecurec.so，请确保系统中存在并能找到上述动态库。
- 接口的生效依赖告警转发守护进程sysalarm，使用时请确保sysalarm进程已启动。
- 使用进程退出后或者不想接收告警时，请调用OS_alarm_UnRegister接口注销。
- 重复多次调用OS_alarm_Register接口注册，均会注册成功，但只有最后一次注册函数会接收到告警消息。

告警注册接收步骤

步骤1 安装支持监控告警特性的操作系统。

步骤2 调用OS_alarm_Register接口注册回调函数。

步骤3 当系统有告警上报时，会调用注册回调函数alarm_callback_func。

步骤4 进程退出或者不需要接收告警时，调用OS_alarm_UnRegister注销告警。

----结束

接口使用参考

表 3-16 alarm_callback_func

接口定义	typedef int (*alarm_callback_func)(void *palarm);
接口描述	告警注册回调函数格式
参数	无须关注参数palarm的具体结构，告警信息被封装在该结构中。
返回值	暂未使用
注意事项	直接传参数palarm到其他接口(见表7-表12)，即可获取详细告警信息

表 3-17 OS_alarm_Register

接口定义	int OS_alarm_Register(struct alarm_hook* alarm);
接口描述	告警注册接口
参数	struct alarm_hook
正常输出	0：成功
异常输出	-1：失败

注意事项	告警接口注册，通过注册回调函数接收alarm上报的告警。在系统中hook只允许注册一次。
	对应的结构体：
	struct alarm_hook{
	alarm_callback_func pfunc_alarmcallbk;
	};

表 3-18 OS_alarm_UnRegister

接口定义	void OS_alarm_UnRegister(void);
接口描述	告警注销接口
参数	无
输出	无
注意事项	进程退出注意注销回调

表 3-19 alarm_report_func

接口名称	typedef int (*alarm_report_func)(unsigned short, unsigned char, unsigned char, unsigned char*)
接口描述	告警注册回调函数格式
参数	参见“告警消息组成”
返回值	暂未使用
注意事项	ucParasLen最大值为255

表 3-20 OS_HookRegister

接口名称	OS_HookRegister
接口描述	告警注册接口
参数	alarm_hook_struct
返回值	0：成功，-1：失败
注意事项	进程退出注意注销回调

表 3-21 OS_UnHookRegister

接口名称	OS_UnHookRegister

接口描述	告警注销接口
参数	无
返回值	无
注意事项	进程退出注意注销回调

表 3-22 OS_alarm_getid

接口定义	unsigned short OS_alarm_getid(void *palarm);
接口描述	获取告警id接口
参数	void *palarm
正常输出	非0：告警ID
异常输出	0：获取不到告警ID信息
注意事项	直接传入palarm参数，无须关注palarm的具体内容

表 3-23 OS_alarm_gettype

接口定义	unsigned short OS_alarm_gettype(void *palarm);
接口描述	获取告警类型接口
参数	void *palarm
输出	告警类型，0：告警恢复，1：告警产生，2：事件发生
注意事项	直接传入palarm参数，无须关注palarm的具体内容

表 3-24 OS_alarm_getlevel

接口定义	unsigned short OS_alarm_getlevel(void *palarm);
接口描述	获取告警级别接口
参数	void *palarm
正常输出	告警级别，1：紧急，2：重要，3：次要，4：提示，5：不确定
异常输出	0：获取不到告警级别
注意事项	直接传入palarm参数，无须关注palarm的具体内容

表 3-25 OS_alarm_gettime

接口定义	long long OS_alarm_gettime(void *palarm);
接口描述	获取告警时间接口
参数	void *palarm
正常输出	long long类型的正整数，表示UTC时间，单位毫秒（ms）
异常输出	0：时间存在异常
注意事项	直接传入palarm参数，无须关注palarm的具体内容

表 3-26 OS_alarm_getdesc

接口定义	char * OS_alarm_getdesc(void *palarm);
接口描述	获取描述信息接口
参数	void *palarm
正常输出	字符串，最长为255个字符
异常输出	NULL
注意事项	直接传入palarm参数，无须关注palarm的具体内容

表 3-27 OS_alarm_getexdesc

接口定义	char * OS_alarm_getexdesc(void *palarm);
接口描述	获取拓展描述信息接口
参数	void *palarm
输出	字符串，最长为255个字符，可以为NULL
注意事项	直接传入palarm参数，无须关注palarm的具体内容

示例代码

以上有两套注册接口，两套目前都支持，第一套注接口参考用例如下：

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include "alarm/register.h"
int callback(void *palarm)
{
    int AlarmId, AlarmLevel, AlarmType;
    long long AlarmTime;
    char *pucParas;
    char *pucExParas;
    FILE *fp;
```



```
    fp = fopen("/var/log/demo.log", "a");

    AlarmId = OS_alarm_getid(palarm);//获取告警id
    AlarmLevel = OS_alarm_getlevel(palarm);//获取告警级别
    AlarmType = OS_alarm_gettype(palarm);//获取告警类型
    AlarmTime = OS_alarm_gettime(palarm);//获取告警时间
    pucParas = OS_alarm_getdesc(palarm);//获取告警信息描述
    pucExParas = OS_alarm_getexdesc(palarm);//获取拓展信息描述

    printf("[alarmid:%d] [alarmlevel:%d] [alarmtype:%d] [alarmtime:%lld ms] [msg:%s] [exmsg:%s]
\n", AlarmId, AlarmLevel, AlarmType, AlarmTime, pucParas, pucExParas);

    return 0;
}

int main()
{
    struct alarm_hook hooker;
    hooker.pfunc_alarmcallbk = callback;
    OS_alarm_Register(&hooker);

    sleep(600);
    OS_alarm_UnRegister();
    return 0;
}
编译命令如下：
gcc register.c -I/usr/include -L/usr/lib64 -lregalarm -o register
```

第二套注册接口，参考用例如下：

```
#include "alarm/register.h"
//告警回调函数实现
int callback(unsigned short usAlarmId, unsigned char ucAlarmType,
         unsigned char ucParasLen, unsigned char* pucParas){
    printf("alarmId:%d alarmType:%d parasLen:%d paras:%s\n",
         usAlarmId, ucAlarmType, ucParasLen, pucParas);
}
int main(int argc, char* argv[]){
    struct alarm_hook_struct hooker;
    hooker.pfunc_alarmreport = callback;
    //注册告警接口
    OS_HookRegister(&hooker);

    sleep(100);
    //注销告警接口
    OS_UnHookRegister();
}
编译命令如下：
gcc register.c -I/usr/include -L/usr/lib64 -lregalarm -o register
```

说明

第一套接口提供给FusionSphere产品使用，第二套接口提供给CRSP产品使用

注册告警 Python 接口

- 注册告警Python接口定义
  - osalarmreg(callfunc)
  - osalarmunreg()
  - getid(palarm)
  - getlevel(palarm)
  - gettype(palarm)
  - gettime(palarm)
  - getdesc(palarm)

- getexdesc(palarm)

注意事项

- Python语言注册接口在regalarm.py模块中实现，请确保系统中存在并能找到该Python模块。
- 接口的生效依赖告警转发守护进程sysalarm，使用时请确保sysalarm进程已启动。
- 使用进程退出后或者不想接收告警时，请调用osalarmunreg接口注销。
- 重复多次调用osalarmreg接口注册，均会注册成功，但只有最后一次注册函数会接收到告警消息。

告警注册接收步骤

步骤1 安装支持监控告警特性的操作系统。

步骤2 调用osalarmreg接口注册回调函数。

步骤3 当系统有告警上报时，会调用注册的回调函数callfunc。

步骤4 进程退出或者不需要接收告警时，调用osalarmunreg注销告警注册。

----结束

接口使用参考

表 3-28 osalarmreg

接口定义	osalarmreg(callfunc)
接口描述	告警注册接口
参数	callfunc[IN]：回调函数的函数名
正常输出	0：成功
异常输出	-1：失败
注意事项	告警接口注册，通过注册回调函数接受alarm上报的告警，在一个系统中只允许注册一次。 告警回调函数的原型为： callfunc(palarm) 其中palarm为告警信息class，保存对应的告警信息，用下面的接口获取各个信息。

表 3-29 osalarmunreg

接口定义	osalarmunreg()
接口描述	告警注销接口
参数	无

输出	无
注意事项	进程退出注意注销回调

表 3-30 getid

接口定义	getid(palarm);
接口描述	获取告警id接口
参数	palarm：告警回调接口传入的告警信息class
正常输出	非0：告警id
异常输出	0：获取不到告警ID信息
注意事项	直接传入palarm参数，无须关注palarm的具体内容

表 3-31 gettype

接口定义	gettype(palarm);
接口描述	获取告警类型接口
参数	palarm：告警回调接口传入的告警信息class
输出	int型，告警类型，0：告警恢复，1：告警产生，2：事件发生
注意事项	直接传入palarm参数，无须关注palarm的具体内容

表 3-32 getlevel

接口定义	getlevel(palarm);
接口描述	获取告警级别接口
参数	palarm：告警回调接口传入的告警信息class
正常输出	int型，告警级别，1：紧急，2：重要，3：次要，4：提示，5：不确定
异常输出	0：获取不到告警级别
注意事项	直接传入palarm参数，无须关注palarm的具体内容

表 3-33 gettime

接口定义	gettime(palarm);

接口描述	获取告警时间接口
参数	palarm：告警回调接口传入的告警信息class
正常输出	long类型的整数，表示UTC时间，单位毫秒（ms）
异常输出	0：时间存在异常
注意事项	直接传入palarm参数，无须关注palarm的具体内容

表 3-34 getdesc

接口定义	getdesc(palarm);
接口描述	获取描述信息接口
参数	palarm：告警回调接口传入的告警信息class
正常输出	字符串，最长为255个字符
异常输出	none
注意事项	直接传入palarm参数，无须关注palarm的具体内容

表 3-35 getexdesc

接口定义	getexdesc(palarm);
接口描述	获取拓展描述信息接口
参数	palarm：告警回调接口传入的告警信息class
输出	字符串，最长为255个字符，可为none
注意事项	直接传入palarm参数，无须关注palarm的具体内容

示例代码

```
import time
import regalarm

def mycallback(palarm):
    f = file('/var/log/pydemo.log','a')
    alarm_info = '[id:%d] [level:%d] [type:%d] [time:%ld] [desc:%s] [exdesc:%s]\n'\
    %(regalarm.getid(palarm),regalarm.getlevel(palarm),regalarm.gettype(palarm),\
    regalarm.gettime(palarm),regalarm.getdesc(palarm),regalarm.getexdesc(palarm))

    f.write(alarm_info)
    f.close()
    return 0

def main():
    ret = regalarm.osalarmreg(mycallback)
    if ret != 0:
        print "reg error"
    else:
```

```
        print "reg success"

    time.sleep(600)
    regalarm.osalarmunreg()

if __name__ == '__main__':
    main()
```

运行命令如下：

```
python register.py
```

3.2.3 如何重发告警

简介

告警模块提供告警重发接口，调用该接口，告警服务会重新上报内部缓存的告警状态。调用停止重发告警接口，告警模块会停止发送告警。

告警注册接口依赖sysalarm服务，可以通过以下命令去启动、关闭、重启sysalarm服务：

```
systemctl start|stop|restart sysalarm
```

重发告警 C 语言接口

- 重发告警C语言接口定义

```
/*****************************************************************
Function     : OS_alarm_resubmit
Description  : 触发告警信息重新上报 该接口为阻塞模式，当告警全部上报完成，才返回，当调用者进程出现异常终止或退出，会停止告警重新上报
Input        : flag [IN]：告警上报类型标记
                  0 全部告警重新上报
                  1 产生告警重新上报
                  2 恢复告警重新上报
Output       : 无
Return       : 0:成功 -1:失败 1:resubmit正在执行 2:停止resubmit命令下发
*****************************************************************/
int OS_alarm_resubmit(int flag);
```

- 重发告警shell接口定义

```
resubmitalarm
usage: resubmitalarm [-f flag]
       -f  flag
           0: all of alarm
           1: generated alarm
           2: resumed alarm
```

- 重发告警python接口定义

```
- osalarmresubmit(flag)
```

停止重发告警接口

- 停止重发告警C语言接口定义

```
/*****************************************************************
Function     : OS_alarm_resubmit_stop
Description  : 触发告警信息重新上报
Input        : 无
Output       : 无
Return       : 0:成功 -1:失败
*****************************************************************/
int OS_alarm_resubmit_stop();
```

- 停止重发告警shell接口定义

```
stopresubmit
```


- 停止重发告警python接口定义
```
- osstopresubmit()
```

注意事项

- C语言注册接口在libregalarm.so动态库中实现，libregalarm.so依赖libsecurec.so，请确保系统中存在并能找到上述动态库。
- 接口的生效依赖告警转发守护进程sysalarm，使用时请确保sysalarm进程已启动。
- 重复多次调用OS_alarm_resubmit接口，如果前一次重发没有结束，再次调用重发接口会返回失败。调用重发接口的进程如果发生异常，进程退出，重发告警也会结束。
- 使用shell接口需要resubmitalarm，stopresubmit文件，存放在/usr/bin下面。
- python接口在regalarm.py模块中实现，请确保系统中存在并能找到该python模块。

3.2.4 如何处理告警

3.2.4.1 磁盘分区异常告警

描述

当磁盘空间利用率超出告警阀值时，系统产生告警。当磁盘空间利用率降低到恢复阀值以下，系统产生恢复告警。

属性

表 3-36 告警属性

告警名称	磁盘分区异常告警
告警ID	1003
告警级别	2（重要）
告警类别	告警产生：1；告警恢复：0
告警时间	接收到告警的时间（UTC时间，单位ms）
告警描述信息	磁盘空间告警：disk partition=%s disk usage=%d 磁盘空间恢复：disk partition=%s disk usage=%d **说明** %s为磁盘分区挂载目录，如/var/log等。 %d为磁盘当前空间占用率。
告警拓展信息	发生异常的磁盘分区目录，如var/log

系统影响

- 如果是根分区满可能导致系统运行异常。
- 如果是日志分区满导致日志丢失，日志转储失败等。

可能原因

- 磁盘空间利用率过高。

处理建议

如果是恢复告警可不处理。如果是产生告警，那么建议按需删除对应磁盘分区下的部分文件以释放空间。

3.2.4.2 ext3/ext4 文件系统故障告警

描述

ext3/ext4文件系统IO处理异常时，系统产生此告警。

属性

表 3-37 告警属性

告警名称	ext3/ext4文件系统异常
告警ID	1004
告警级别	2（重要）
告警类别	告警产生：1
告警时间	接收到告警的时间（UTC时间，单位ms）
告警描述信息	文件系统只读：*dev* filesystem error. Remount filesystem read-only 其他异常：*dev* filesystem error **说明** dev为磁盘分区，如sda1等。
告警拓展信息	发生异常的磁盘分区名称，如sda1

系统影响

- 根分区只读会导致该服务器上大部分服务进程异常，虚拟机出现严重故障。
  日志分区会被错误日志填满，日志无法及时转储，部分日志可能丢失。
- /var/log分区只读会导致日志写入，日志转储等相关操作失败。

可能原因

- 服务器异常掉电。
- 服务器磁盘硬件损坏。
- 服务器RAID卡硬件损坏 。
- 磁盘驱动版本与硬件不匹配。
- 若为非本地存储，主机与存储之间连接断开。

- 磁盘数据损坏。

处理建议

步骤1 检查上报告警后主机是否重启过。

- 是，若能正常进入系统，请手动清除告警；否则请执行**步骤5**
- 否，请执行**步骤2**

步骤2 查看上报的告警信息。

- 若为“xxx filesystem error. Remount filesystem read-only”，请执行**步骤3**。
- 若为“xxx filesystem error”，请执行**步骤5**。

步骤3 运行以下命令，查看只读的磁盘分区中是否有告警信息中的磁盘分区。

```
cat /proc/mounts |grep "ro,"
```

查看命令是否有输出结果：

命令回显如下所示（第一列为磁盘分区）：

```
tmpfs /sys/fs/cgroup tmpfs ro,nosuid,nodev,noexec,mode=755 0 0
/dev/sda1 /boot ext3 ro,relatime,errors=continue,user_xattr,acl,barrier=1,data=ordered 0 0
```

- 是（若告警信息中也有/dev/sda1），请执行**步骤4**。
- 否，请手动清除告警。

步骤4 查看只读的磁盘分区（如步骤三中的/dev/sda1）是否为用户手动挂载的只读。

- 是，请手动清除告警。
- 否，请执行**步骤6**。

步骤5 联系服务器硬件厂商，检查磁盘或RIAD卡是否损坏。

说明

如果服务器磁盘或者RIAD卡损坏，请考虑迁移磁盘数据至其他服务器。

- 是，请更换硬件，重启单板。
- 否，请执行**步骤6**。

步骤6 请联系华为技术支持。

----结束

3.2.4.3 关键进程异常告警

描述

当监控的进程或服务状态异常或者异常恢复时，系统产生此告警。

属性

表 3-38 告警属性

告警名称	关键进程异常
告警ID	1005

告警级别	2（重要）
告警类别	告警产生：1， 告警恢复：0
告警时间	接收到告警的时间（UTC时间，单位ms）
告警描述信息	告警产生：%s is abnormal 告警恢复：%s is recovered **说明** %s为监控的进程名，如sshd等。
告警拓展信息	发生异常的进程名称，如crond， sshd等

系统影响

- 可能导致系统异常。
- 可能导致业务功能异常。

可能原因

- 进程异常终止。
- 服务异常。

处理建议

步骤1 根据告警信息获取告警的类别，如果是告警恢复可不处理。

步骤2 若告警类别是异常产生，获取异常的进程或服务。

步骤3 根据/etc/sysmonitor/process/下相关配置文件配置的RECOVER_COMMAND，尝试恢复异常的进程或服务。

步骤4 根据/etc/sysmonitor/process/下相关配置文件配置的MONITOR_COMMAND，查看进程或服务是否恢复正常

- 是，执行完毕。
- 否，执行步骤5。

步骤5 请联系华为技术支持处理。

----结束

3.2.4.4 文件异常告警

描述

当指定的文件被删除，或者增加/删除子文件、增加/删除子目录时，系统产生此告警。

属性

表 3-39 告警属性

告警名称	文件监控事件发生
告警ID	1006
告警级别	5（不确定）
告警类别	事件发生：2
告警时间	接收到告警的时间（UTC时间，单位ms）
告警描述信息	目录下增加子文件：Subfile %s under %dir was added 目录下删除子文件：Subfile %s under %dir was deleted 目录下删除子目录：Subdir %s under %dir was added 目录下删除子目录：Subdir %s under %dir was deleted 文件被删除：File %s was deleted **说明** %s为增加或删除的文件、目录名，%dir表示被监控的目录。
告警拓展信息	添加或删除的文件或文件夹

系统影响

- 文件大量增加会导致系统分区占用率增加，也可能说明是某些业务异常。
- 关键文件被删除会影响业务甚至系统运行。

可能原因

- 人为操作
- 程序操作

处理建议

步骤1 查看详细文件监控告警信息，根据发生事件和对应文件来判断是否影响系统运行。

- 否，执行完毕。
- 是，执行步骤2。

步骤2 请联系华为技术支持处理。

----结束

3.2.4.5 网卡状态异常告警

描述

当指定网卡发生up、down或者新增删除IP地址时，系统产生此告警。

属性

表 3-40 告警属性

告警名称	网卡异常
告警ID	1007
告警级别	5（不确定）
告警类别	事件发生：2
告警时间度	接收到告警的时间（UTC时间，单位ms）
告警描述信息	*netcard*: device is up/down *netcard*: ip[*xxx.xxx.xxx.xxx*] is added/deleted **说明** netcard为网卡名，如eth1等；xxx.xxx.xxx.xxx为点分形式的IP地址。
告警拓展信息	对应的网卡名

系统影响

- 网络可能发生异常。

可能原因

- 人为或程序操作。

处理建议

步骤1 通过PUTTY登录主机，观察配置网卡是否在ifconfig命令输出的列表中？

- 是，执行完毕。
- 否，执行**步骤2**。

步骤2 使用ifconfig命令手动恢复对应网卡。

```
#ifconfig eth0 up
```

----结束

3.2.4.6 信号异常告警

描述

当给进程发指定信号时，系统产生此告警。

属性

表 3-41 告警属性

告警名称	信号异常告警

告警ID	1008
告警级别	5（不确定）
告警类别	事件发生：2
告警时间	接收到告警的时间（UTC时间，单位ms）
告警描述信息	*process1*[*pid1*] send *signal* to *process2*[*pid2*] **说明** process1为发信号的进程，pid1为发信号进程的进程号，signal为发送的信号（如SIGKILL），process2为收信号的进程，pid2为收信号的进程号。
告警拓展信息	发送信号的进程/模块名称及进程号

系统影响

- 业务进程可能被异常终止。

可能原因

- 人为操作原因被kill。
- 进程异常。
- 系统异常。

处理建议

步骤1 通过PUTTY登录主机，通过查看sysmonitor.log日志，查询发送信号的进程判断进程被杀是否正常。

- 是，执行完毕。
- 否，执行**步骤2**。

步骤2 请联系华为技术支持处理。

----结束

3.2.4.7 CPU 异常告警

描述

当cpu占用率过高时，系统产生此告警。

属性

表 3-42 告警属性

告警名称	cpu占用率异常告警
告警ID	1001

告警级别	2（重要）
告警类别	告警产生：1， 告警恢复：0
告警时间度	接收到告警的时间（UTC时间，单位ms）
告警描述信息	告警描述信息：CPU usage alarm: 4.1f% 告警恢复信息：CPU usage resume: 4.1f% **说明** 4.1f%为告警时的cpu占用率。
告警拓展信息	暂无

系统影响

- cpu使用率过高，业务进程可能调度不到。

可能原因

- 系统负载过大。
- 系统异常。

处理建议

步骤1 通过PUTTY登录主机，通过top命令查看cpu占用情况是否正常？

- 是，执行完毕。
- 否，执行**步骤2**。

步骤2 请联系华为技术支持处理。

----结束

3.2.4.8 内存异常告警

描述

当内存占用率过高时，系统产生此告警。

属性

表 3-43 告警属性

告警名称	内存占用率异常告警
告警ID	1002
告警级别	2（重要）
告警类别	告警产生：1， 告警恢复：0
告警时间	接收到告警的时间（UTC时间，单位ms）

告警描述信息	告警描述信息：memory usage alarm: 4.1f% 告警恢复信息：memory usage resume: 4.1f% **说明** 4.1f%为告警时的内存占用率。
告警拓展信息	暂无

系统影响

- 内存占用率过高，业务进程可能运行异常。

可能原因

- 系统负载过大。
- 系统异常。

处理建议

步骤1 通过PUTTY登录主机，通过free命令查看内存占用情况是否正常？

- 是，执行完毕。
- 否，执行**步骤2**。

步骤2 请联系华为技术支持处理。

----结束

3.2.4.9 进程数异常告警

描述

当系统进程数量过高时，系统产生此告警。

属性

表 3-44 告警属性

告警名称	进程数目异常告警
告警ID	1009
告警级别	3（次要）
告警类别	告警产生：1， 告警恢复：0
告警时间	接收到告警的时间（UTC时间，单位ms）
告警描述信息	告警描述信息：process count alarm: %ld 告警恢复信息：process count resume: %ld **说明** %ld为告警时系统的进程数。

告警拓展信息	暂无

系统影响

进程数过高，系统可能无法创建新的进程。

可能原因

- 系统异常。

处理建议

步骤1 通过PUTTY登录主机，通过ps命令查看进程情况是否正常？

- 是，执行完毕。
- 否，执行**步骤2**。

步骤2 请联系华为技术支持处理。

----结束

3.2.4.10 系统句柄总数异常告警

描述

当系统fd数量过高时，系统产生此告警。

属性

表 3-45 告警属性

告警名称	fd数目异常告警
告警ID	1010
告警级别	3（次要）
告警类别	告警产生：1， 告警恢复：0
告警时间	接收到告警的时间（UTC时间，单位ms）
告警描述信息	告警描述信息：fd count alarm: %lu 告警恢复信息：fd count resume: %lu **说明** %lu为告警时系统的fd总数。
告警拓展信息	暂无

系统影响

文件句柄数量过高，系统可能无法打开新的文件或者套接字。

可能原因

- 系统异常。

处理建议

步骤1 通过PUTTY登录主机，通过lsof命令查看系统fd占用情况是否正常？

- 是，执行完毕。
- 否，执行步骤2。

步骤2 请联系华为技术支持处理。

----结束

3.2.4.11 单个进程句柄数告警

描述

当系统单个进程fd数量过高时，系统产生此告警。

属性

表 3-46 告警属性

告警名称	单个进程句柄数告警
告警ID	1014
告警级别	5（不确定）
告警类别	事件发生：2
告警时间	接收到告警的时间（UTC时间，单位ms）
告警描述信息	cmdline [%s] pid %d, fd number %u, more than %u **说明** 第一个变量为命令执行路径及参数，第二个变量为pid，第三个变量为当前进程的句柄数，第四个变量为事件发生的阈值。
告警拓展信息	暂无

系统影响

单个进程文件句柄数量过高，该进程可能无法打开新的文件或者套接字。

可能原因

- 系统异常。
- 句柄泄露

处理建议

步骤1 通过PUTTY登录主机，通过lsof命令查看告警信息中的进程fd占用情况是否正常。

- 是，执行完毕。
- 否，执行步骤2。

步骤2 请联系华为技术支持处理。

----结束

3.2.4.12 磁盘 inode 异常告警

描述

当磁盘inode利用率超出告警阀值时，系统产生告警。当磁盘inode利用率降低到恢复阀值以下，系统产生恢复告警。

属性

表 3-47 告警属性

告警名称	磁盘分区异常告警
告警ID	1011
告警级别	2（重要）
告警类别	告警产生：1；告警恢复：0
告警时间	接收到告警的时间（UTC时间，单位ms）
告警描述信息	inode告警产生：disk partition=%s inode usage=%d inode告警恢复：disk partition=%s inode usage=%d **说明** %s为磁盘分区挂载目录，如/var/log等。 %d为磁盘当前inode使用率。
告警拓展信息	发生异常的磁盘分区目录，如var/log

系统影响

- 如果是根分区满可能导致系统运行异常。
- 如果是其它分区满会导致无法创建文件等。

可能原因

- 磁盘inode利用率过高。

处理建议

如果是恢复告警可不处理。如果是产生告警，那么建议按需删除对应磁盘分区下的部分文件以释放inode。

3.2.4.13 存储磁盘 I/O 延时过大告警

描述

每5s中读取一次本地磁盘的I/O时延并保存，每5分钟读取这5分钟内计算的I/O时延平均值，如果该平均值有超过一半（30次以上）大于用户配置的IO延时阀值，则系统产生告警。当有一半及以上小于等于阀值时，系统产生恢复告警。

属性

表 3-48 告警属性

告警名称	存储磁盘I/O时延过大告警
告警ID	1012
告警级别	2（重要）
告警类别	告警产生：1；告警恢复：0
告警时间	接收到告警的时间（UTC时间，单位ms）
告警描述信息	磁盘%sIO时延过大 磁盘%sIO时延正常 **说明** %s磁盘ID，如sda等。
告警拓展信息	故障的磁盘ID号 如：sdb

系统影响

- 导致处理磁盘I/O读写占用的CPU利用率过高。
- CPU处理业务速度较慢。

可能原因

- 硬件原因导致磁盘I/O读写响应过慢。
- 没有组raid导致性能问题，或者raid卡驱动等有问题

处理建议

步骤1 使用如下命令，检查是否支持smart。

```
smartctl
```

- 如果命令回显输出：SMART support is: Enabled ，表示支持smart。执行步骤2。
- 如果命令回显输出：Device does not support SMART ，表示不支持smart。

  不支持smart的情况，一般是因为配置的raid卡不支持，此时需要使用对应raid卡厂商的检查工具。

步骤2 使用smart工具自检，确认是否为硬件问题。

1. 先查看基本的smart信息。
2. 再查看硬盘GLIST列表。
3. 触发smart自检。
4. 如果检查通过，则可认为磁盘暂时无严重问题；如果再出现多次慢盘场景则建议更换硬盘或联系硬盘厂家。

步骤3 请联系华为技术支持处理。

----结束

3.3 监控日志

简介

在默认情况下，为防止sysmonitor.log文件过大，日志特性提供了到2M阈值后，切分转储日志的机制。日志将被转储到磁盘目录下，这样就能够保存一定量的日志。

配置文件说明

配置文件为：/etc/rsyslog.d/sysmonitor.conf，配置文件示例如下：

```
$FileCreateMode 0600
$outchannel sysmonitor, /var/log/sysmonitor.log, 2097152, /usr/libexec/sysmonitor/
sysmonitor_log_dump.sh
if ($programname == 'sysmonitor' and $syslogseverity <= 6) then {
:omfile:$sysmonitor
stop
}

if ($programname == 'sysmonitor' and $syslogseverity > 6) then {
/dev/null
stop
}
```

注意

/usr/libexec/sysmonitor/sysmonitor_log_dump.sh中会调用logrotate /usr/libexec/sysmonitor/sysmonitor-logrotate切割日志文件，若修改日志文件转储大小，请确保/etc/rsyslog.d/sysmonitor.conf中的转储阈值比/usr/libexec/sysmonitor/sysmonitor-logrotate中的size配置值大

3.4 告警日志

简介

在系统/var/log/目录下，存有sysalarm.log，记录着告警模块的程序运行的异常日志和过程日志，并且还记录着告警或事件的上报日志等，可用于问题定位和分析。

为防止sysalarm.log文件过大，日志特性提供了到2M阈值后，切分转储日志的机制。日志将被转储到磁盘目录下，这样就能够保存一定量的日志。

告警特性日志转储配置文件存在于/etc/rsyslog.d/目录下，名为sysalarm.conf。

配置文件说明

配置文件为：/etc/rsyslog.d/sysalarm.conf，配置文件示例如下：

```
$FileCreateMode 0600
$outchannel sysalarm, /var/log/sysalarm.log, 2097152, /usr/libexec/sysalarm/sysalarm_log_dump.sh
if ($programname == 'sysalarm' and $syslogseverity <= 6) then {
:omfile:$sysalarm
stop
}

if ($programname == 'sysalarm' and $syslogseverity > 6) then {
/dev/null
stop
}
```

注意

/usr/libexec/sysalarm/sysalarm_log_dump.sh中会调用logrotate /usr/libexec/sysalarm/sysalarm-logrotate切割日志文件，若修改日志文件转储大小，请确保/etc/rsyslog.d/sysalarm.conf中的转储阈值比/usr/libexec/sysalarm/sysalarm-logrotate中的size配置值大

4 健康检查工具

OS健康检查工具用来检查EulerOS的主要进程运行是否正常、数据及配置文件是否丢失等。技术支持工程师和维护工程师使用该工具，可以降低日常检查难度，保障产品的可维护性。

4.1 简介

介绍健康检查工具的产品定位、 产品功能以及软件获取。

产品定位

OS健康检查工具（osHealthCheck）用来检查EulerOS系统的主要进程运行是否正常、数据及配置文件是否丢失等。

该工具方便技术支持工程师和维护工程师快速了解系统的健康状况。

产品功能

健康检查工具提供的功能如**表4-1**所示。

表 4-1 健康检查工具功能描述

功能	简要描述
检查业务节点的黑匣子功能是否开启	如果未正常运行，则提示异常；否则该项检查正常。
检查系统文件完整性	检查系统关键文件是否被更改。
检查Cgroup服务是否正常	检查Cgroup分区是否正常挂载。
检查printk缓冲区重定向服务是否正常	检查printk缓冲区重定向功能开启后是否正常。

功能	简要描述
检查系统服务是否配置超时	检查各service文件是否配置了超时功能

软件获取

本工具在安装EulerOS版本时自动安装。

4.2 使用说明

介绍健康检查工具使用方法和 工具异常现象处理。

4.2.1 执行系统健康检查

命令原型

```
osHealthCheck  -i <id>
osHealthCheck  -s <id1,id2...>
osHealthCheck  -l
osHealthCheck  -i <id> -p <dirname>
osHealthCheck -s <id1,di2...> -p <dirname>
```

参数说明

参数	说明
-i	按照id 进行检查，不同的id对应不同的检查项。id为整型，且必须在规定范围内[10001, 10002, 10004, 10005, 10006]（具体含义请参考检查项）。
-s	执行[10001, 10002, 10004, 10005, 10006]中一个或多个检查项，不同检查项之间用逗号隔开。
-l	列出所有检查项
-p	将生成的结果文件存放到指定的目录中，该目录需要已经建立。该参数只能与-i或-s一起使用。

注意事项

1. 在执行健康检查时会消耗cpu资源，为避免健康检查影响虚拟机业务，请将osHealthCheck限制在cgroup中运行以控制资源使用。如果osHealthCheck调用者已经限制在cgroup中，则osHealtCheck会继承调用者的cgroup执行环境。
2. 根分区满的情况下会导致某些检查项异常，执行健康检查之前请确保根分区未满。
3. 检查项10002，依赖文件完整性的包，如果该包没有安装，则检查失败。

返回值说明

返回值	含义	处理建议
0	检查生成结果文件。	查看xml结果文件
2	参数传入错误或已有健康检查进程时退出。	按照提示重新执行健康检查工具。
3	/var/log目录满，日志信息无法写入。	请参考**4.2.3 处理工具异常现象**。

示例

以id为10001的检查项为例，使用如下命令执行健康检查工具。

```
osHealthCheck -i 10001 -p /home
```

4.2.2 查看健康检查报告

查看汇总报告

1. 执行健康检查工具后，进入如下目录（或-p参数指定的目录，在业务节点上，该目录由业务端保证其存在）。
   ```
   /var/log/osHealthCheck/
   ```
2. 执行如下命令，根据生成report.xml文件的时间，判断该xml文件是否最新。
   ```
   ll report.xml
   ```
3. 如果该xml文件为最新，则查看该xml文件。报告文件中包含当前已检查的所有项目、检查成功项、检查失败项。xml文件格式如下：
   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <Report>
     <TotalItems>10001,10002,10004</TotalItems>
     <FailedItems>10001</FailedItems>
     <PassedItems>10002,10004</PassedItems>
   </Report>
   ```
4. 如果report.xml未生成，请参考**4.2.3 处理工具异常现象**。

   说明

   在通过osHealthCheck -i检查单个健康检查项时不会生成汇总报告，只会生成<id>_itemresult.xml文件。

查看 xml 文件

1. 执行健康检查工具后，进入如下目录（或-p参数指定的目录，在业务节点上，该目录由业务端保证其存在）。
   ```
   /var/log/osHealthCheck/result
   ```
2. 执行如下命令，根据生成<id>_itemresult.xml文件的时间，判断该xml文件是否最新。
   ```
   ll 10001_itemresult.xml
   ```
3. 如果该xml文件为最新，则查看该xml文件。否则请参考**4.2.3 处理工具异常现象**。

查看日志文件

执行健康检查工具后，可进入如下目录，查看对应的日志文件。如果对应的日志文件为空，请参考**4.2.3 处理工具异常现象**。


```
/var/log/osHealthCheck
```

其中，日志名为osHealthCheck.log。

说明

osHealthCheck.log文件每天会进行切割，若该文件大于2M就切割生成一个同名带数字后缀的文件。切割文件最多保留20份，若超过20份则会删除最早生成的切割文件。该日志文件不支持转储。

4.2.3 处理工具异常现象

调用osHealthCheck健康检查工具，可能会出现如下两个异常情况。

xml 文件异常

异常描述

在如下目录下没有生成最新的xml文件或对应id的xml文件。

```
/var/log/osHealthCheck/result
```

原因分析

- 参数格式错误或者参数范围不正确。
- 存放xml文件的目录没有剩余空间。

处理方法

以下处理建议与上述可能原因中顺序对应。

- 按照正确的参数格式、参数范围输入参数。
- 释放xml文件所在目录空间。

日志文件异常

异常描述

在如下目录下对应时间点的日志文件的内容为空。

```
/var/log/osHealthCheck
```

原因分析

/var/log目录没有剩余空间。

处理方法

释放/var/log目录空间。

4.3 检查项

本主题介绍了健康检查工具的5个检查项，并给出每个检查项的功能、现象描述、可能原因、定位思路以及处理方法。

4.3.1 10001 检查业务节点的黑匣子功能是否开启

检查项功能

检查业务节点的黑匣子功能是否开启。

现象描述

查看XML文件，出现如下所示异常信息。

```
<item>
<id>10001</id>
<name>Kbox function of the service node.</name>                          <!-- 检查项 -->
<result>Abnormal</result>                                               <!-- 检查结果 -->
<output>The kbox function of the service node is disabled.</output>     <!-- 异常信息 -->
</item>
```

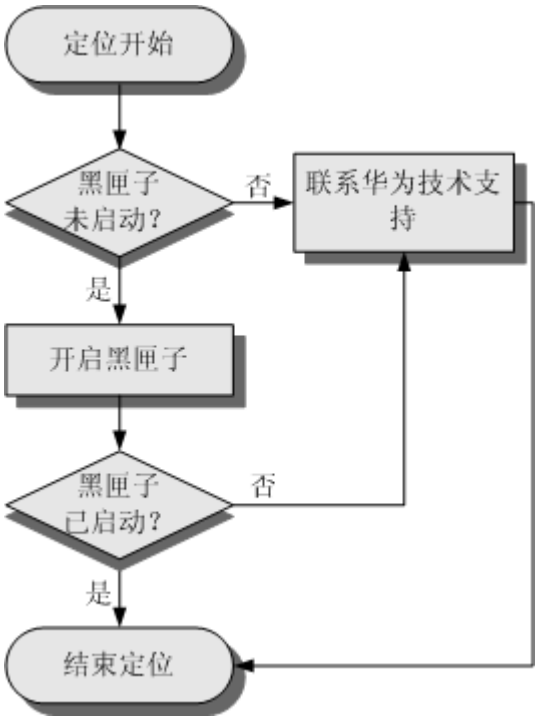
原因分析

黑匣子服务没有开启。

定位思路

故障定位思路如图4-1所示。

图 4-1 故障定位思路流程图



处理步骤

1. 使用PuTTY登录故障节点。
2. 执行如下命令，检查黑匣子是否启动。

   ```
   systemctl status kbox
   ```


查看状态是否为running。

- 是 => 5
- 否 => 3

3. 执行如下命令，启动黑匣子。
```
systemctl start kbox
```
4. 调用健康检查工具，对该检查项重新检查，并查看最新健康检查报告，查看检查结果“result”是否为“OK”。
   - 是 => 处理完毕
   - 否 => 5
5. 请联系华为技术支持处理。

4.3.2 10002 检查系统文件完整性

检查项功能

检查系统文件完整性。

现象描述

查看XML文件，出现如下所示异常信息。

```
<item>
<id>10002</id>
<name>Check integrity of system files.</name>                    <!-- 检查项 -->
<result>Abnormal</result>                                        <!-- 检查结果 --
>
<output>Some system files are abnormal.</output>                 <!-- 异常信息 --
>
</item>
```

说明

执行10002检查项之前请确保根分区未满；否则会导致cp、sed命令失败，从而造成检查结果异常。

原因分析

1. 必要文件被删除。
2. 必要文件被修改。

定位思路

根据日志进行定位。

处理步骤

1. 使用PuTTY登录故障节点。
2. 查看日志。
```
cat /usr/libexec/osfilecheck/result/*
```
在该目录的日志中，记录着删除文件、被修改文件的信息，如图4-2所示。

图 4-2 健康检查日志

```
Euler:/usr/libexec/osfilecheck/result # cat osfilecheck-report2
AIDE 0.15.1 found differences between database and filesystem!!
Start timestamp: 2018-01-10 14:16:15

Summary:
  Total number of files:        505
  Added files:                  0
  Removed files:                1
  Changed files:                0


---------------------------------------------------
Removed files:
---------------------------------------------------

removed: /bin/sleep
```

3. 获取必要文件的修改信息，分析修改原因和影响。
4. 恢复相应文件后调用健康检查工具，对该检查项重新检查，并查看最新健康检查报告，查看检查结果“result”是否为“OK”。
   - 是 => 处理完毕
   - 否 => 5
5. 请联系华为技术支持处理。

4.3.3 10004 检查 Cgroup 服务是否正常

检查项功能

检查Cgroup服务是否正常。

现象描述

查看XML文件，出现如下所示异常信息。

```
<item>
<id>10004</id>
<name>Cgroup function of the service node.</name>                    <!-- 检查
项 -->
<result>Abnormal</result>                                            <!-- 检查
结果 -->
<output>The cgroup function of the service node is abnormal.</output>  <!-- 异常
信息 -->
</item>
```

原因分析

Cgroup分区没有全部挂载。

定位思路

根据mount信息查看cgroup子系统挂载情况。

处理步骤

1. 使用PuTTY登录故障节点。
2. 执行如下命令，检查cgroup分区（blkio,cpu,cpuacct, cpuset, devices,freezer,hugetlb,memory,net_cls,perf_event）是否全部挂载。
   ```
   mount | grep cgroup
   ```
   - cgroup分区没有全部挂载=> 3

     例如图4-3回显信息中缺少blkio分区，表示bikio分区没有挂载。

     图 4-3 bikio 分区未挂载

     ```
     cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
     cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
     cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
     cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpuacct,cpu)
     cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
     cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
     cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)
     cgroup on /sys/fs/cgroup/net_cls type cgroup (rw,nosuid,nodev,noexec,relatime,net_cls)
     ```

   - cgroup分区全部挂载 => 5

     图4-4回显信息表示分区全部挂载。

     图 4-4 分区全部挂载

     ```
     cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
     cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
     cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
     cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpuacct,cpu)
     cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
     cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
     cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio)
     cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)
     cgroup on /sys/fs/cgroup/net_cls type cgroup (rw,nosuid,nodev,noexec,relatime,net_cls)
     ```

3. 执行如下命令，重新挂载cgroup子系统。

   例如图4-3中blkio分区没有挂载，执行如下命令重新挂载。
   ```
   mount -t cgroup -o blkio cgroup /sys/fs/cgroup/blkio
   ```
4. 调用如下命令对该检查项重新检查，
   ```
   osHealthCheck -i 10004
   ```
   查看最新健康检查报告，查看检查结果“result”是否为“OK”。
   - 是 => 处理完毕。
   - 否 => 5。
5. 请联系华为技术支持处理。

4.3.4 10005 检查 printk 缓冲区重定向服务是否正常

检查项功能

检查printk缓冲区重定向服务配置后是否正常。

现象描述

查看XML文件，出现如下所示异常信息。


```
<item>
<id>10005</id>
<name>Printk buffer redirect function of the service node.</name>
<result>Abnormal</result>
<output>The printk buffer redirect function enable, but not working.</output>
</item>
```

原因分析

重定向服务配置项已经配置，但是配置未生效或者配置不正确。

定位思路

缓冲区重定向功能属于调测特性，依赖非遗失内存介质（系统reboot复位，内存不清0），如果当前系统中无这类介质，该功能不能使能，无需排查。该特性是非必须特性。

处理步骤

针对已经使能的系统，排查步奏如下：

1. 使用PuTTY登录故障节点。
2. 执行如下命令，检查重定向服务是否启动。
   ```
   systemctl status redirect
   ```
   查看状态是否为active。
   - 是 => 5
   - 否 => 3
3. 执行如下命令，重启重定向功能。
   ```
   systemctl restart redirect
   ```
4. 调用健康检查工具，对该检查项重新检查，并查看最新健康检查报告，查看检查结果“result”是否为“OK”。
   - 是 => 处理完毕
   - 否 => 5
5. 请联系华为技术支持处理。

4.3.5 10006 检查系统服务是否配置超时

检查项功能

检查系统各service文件是否配置超时功能。

现象描述

查看XML文件，出现如下所示异常信息。

```
<item>
<id>10006</id>
<name>Check timeout of services.</name>                                    <!-- 检查项 -->
<result>Abnormal</result>                                                  <!-- 检查结果 -->
<output>19 services are not configured timeout.</output>                   <!-- 异常信息 -->
</item>
```

原因分析

1. 相关service文件配置了TimeoutSec=0。
2. 相关service文件配置了TimeoutStartSec=0。
3. 相关service文件配置了TimeoutStopSec=0

定位思路

根据日志进行定位。

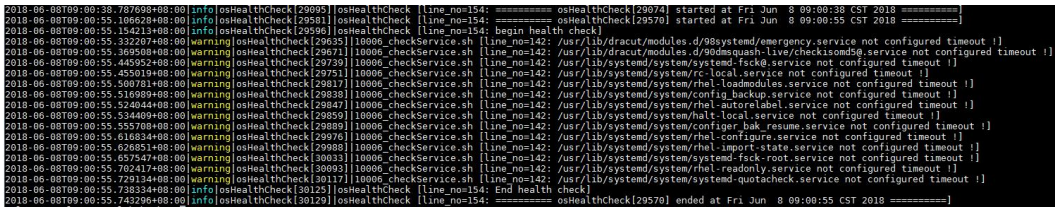
处理步骤

1. 使用PuTTY登录故障节点。
2. 查看日志。

   ```
   cat /var/log/osHealthCheck/osHealthCheck.log
   ```

   在该目录的日志中，记录着没有配置超时的系统服务信息，如图4-5所示。

   图 4-5 健康检查日志

3. 查看对应service文件，获取必要文件的修改信息，分析修改原因和影响。
4. 修改对应的service文件，配置超时功能，对该检查项重新检查，并查看最新健康检查报告，查看检查结果“result”是否为“OK”。
   - 是 => 处理完毕
   - 否 => 5
5. 请联系华为技术支持处理。

5 日志防爆特性

5.1 功能简介

日志防爆特性，是指系统会通过日志切割工具logrotate ，每隔一小时对/etc/logrotate.d目录下的日志配置文件中配置的日志文件进行切割，以防止磁盘空间被占满。

默认已添加的日志切割文件列表有：

表 5-1 日志切割文件列表

日志名称
/var/log/cron
/var/log/maillog
/var/log/messages
/var/log/secure
/var/log/spooler
/var/log/yum.log
/var/log/startup.log
/var/log/boot.log
/var/log/backup_conf.log
/var/log/cloud-init-output.log
/var/log/cloud-init.log
/var/log/daemon.log
/var/log/cron.log
/var/log/kern.log
/var/log/syslog
/var/log/unused.log
/var/log/tuned/*.log
/var/log/wtmp
/var/log/btmp
/var/log/firewalld

注意

其它组件新增的日志文件根据需要添加相应的切割配置，具体使用方法可参见**5.3 使用方法**。

5.2 使用场景

系统使用过程中，日志持续增长会使存储日志的磁盘空间被占满，从而导致日志无法继续打印，或者影响到系统部分业务功能（进程异常或一些操作无法进行）；另外如果系统出现了问题，因为缺少日志也会影响问题定位效率，甚至导致问题无法定位。

日志防爆特性，主要是针对大小会持续增长的日志文件，对其添加切割配置，确保存储日志文件磁盘空间不被占满，从而增强系统的稳定性与可靠性。

5.3 使用方法

在/etc/logrotate.d目录下，对需要切割的日志文件进行配置。

文件配置方法

在/etc/logrotate.d目录下，添加配置文件（新增配置文件的权限建议不大于640），一个配置文件中可同时配置多个需要切割的日志。

如：/etc/logrotate.d/example文件中配置。

```
/var/log/logexample
/var/log/logexample1
{
maxage 365
rotate 30
notifempty
compress
copytruncate
missingok
size +4096k
}
```

该配置会对/var/log/logexample、/var/log/logexample1进行切割。

- maxage 365：只存储最近365天的切割出来的日志文件，超过365天则删除。
- rotate 30：指定日志文件删除之前切割的次数，此处保留30个备份。
- notifempty：表示日志为空则不处理。
- compress：通过gzip压缩转储以后的日志。
- copytruncate：用于还在打开中的日志文件，把当前日志备份并截断。
- missingok：如果日志文件丢失，不报错继续执行下一个。
- size +4096k：表示日志超过4096k大小才分割，size默认单位是KB，可使用k、M和G来指定KB、MB和GB。

配置项说明

配置项说明如**表1**。

表 5-2 配置项说明

配置项	功能
compress	通过gzip压缩转储以后的日志。
missingok	找不到日志时，跳过。
nomissingok	找不到日志时，报错。
nocompress	不需要压缩时，用这个参数。
copytruncate	用于还在打开中的日志文件，把当前日志备份并截断。
nocopytruncate	备份日志文件但是不截断。
create mode owner group	转储文件，使用指定的文件模式创建新的日志文件。
nocreate	不建立新的日志文件。
prerotate/endscript	在转储以前需要执行的命令可以放入这个对，这两个关键字必须单独成行。

配置项	功能
postrotate/endscript	在转储以后需要执行的命令可以放入这个对，这两个关键字必须单独成行。
daily	指定转储周期为每天。
weekly	指定转储周期为每周。
monthly	指定转储周期为每月。
rotate count	指定日志文件删除之前转储的次数，0指没有备份，5指保留5个备份。
size	当日志文件到达指定的大小时才转储，size可以指定bytes（缺省）以及KB、MB或者GB。

配置项中“转储”的含义：用来把旧的日志文件删除，并创建新的日志文件。

说明

1. nocreate与配置文件中的copytruncate是互斥的，不能同时配置，否则nocreate不生效。
2. create mode owner group（例如：create 0600 root root）与配置文件中的copytruncate是互斥的，否则create配置不生效。
3. 时间频度（daily，weekly，monthly，yearly）和日志大小（size）这两项参数同时配置的时候，会以日志大小为条件，达到一定大小就会切割。

