内核软死锁处理方案

一、问题背景

服务器：海光服务器

系统版本：xxx_4.19_.x86_64

【问题复现步骤】

插入 xxx_lru_xxx.ko 模块：insmod xx/yy/xxx_lru_xx.ko

（自研模块查看 LRU 链表等相关信息）

调用 dump 接口，打印 LRU 链表信息

echo dump > /proc/yy/xxx_lru_xxx

【问题现场】

调用接口后，系统软锁死告警，系统随后 crash 复位

kernel:[xxx] watchdog: BUG: soft lockup - CPU#y stuck for 22s! [bash:yy]

查看 proc 下对 watchdog 配置

watchdog_print_period: 10 - 打印看门狗信息时间间隔

watchdog_softlockup_divide: 5 - 软死锁检测的间隔时间

watchdog_thresh:10 - 看门狗超时阈值，超时没有收到喂狗信息就会触发重启

并且 cmdline 中已配置 softlockup_panic=1 看门狗使能

二、问题分析定位

1、 crash 分析

根据系统复位后转储的 vmcore 文件，使用 crash 工具进行分析

crash vmlinux vmcore (vmlinux 和 当前系统版本信息一致)

查看 crash 时候的调用栈，RIP 在 console_unlock+801 处，后续

apic_timer_interrupt 流程是狗叫的流程。（RIP - 指令指针寄存器，指向导致故障或异常的 CPU 指令）

```
crash> bt
PID: 2970   TASK: ffff8f4448ff1780  CPU: 5   COMMAND: "bash"
 #0 [ffff8f447fd43d50] machine_kexec at ffffffff81a55dcf
 #1 [ffff8f447fd43da8] __crash_kexec at ffffffff81b59b91
 #2 [ffff8f447fd43e68] panic at ffffffff81aaf225
 #3 [ffff8f447fd43ef0] watchdog_timer_fn at ffffffff81b8ddf4
 #4 [ffff8f447fd43f20] __hrtimer_run_queues at ffffffff81b397a8
 #5 [ffff8f447fd43f80] hrtimer_interrupt at ffffffff81b39f95
 #6 [ffff8f447fd43fd8] smp_apic_timer_interrupt at ffffffff824025aa
 #7 [ffff8f447fd43ff0] apic_timer_interrupt at ffffffff82401b0f
--- <IRQ stack> ---
 #8 [fffffb3a080b27ab8] apic_timer_interrupt at ffffffff82401b0f
    [exception RIP: console_unlock+801]
    RIP: ffffffff81b17dd1  RSP: fffffb3a080b27b68  RFLAGS: 00000246
    RAX: 0000000000000001  RBX: ffffffff834be290  RCX: 00000000ffffffff
    RDX: 0000000000000000  RSI: 0000000000000004  RDI: 0000000000000246
    RBP: 000000000000004e   R8: 0000000000000000   R9: 0000000000aaaaaa
    R10: 0000000000000001  R11: 00000000ffffffff  R12: 0000000000000000
    R13: ffffffff834bb64c  R14: ffffffff82e678a0  R15: 0000000000000000
    ORIG_RAX: ffffffffffffff13  CS: 0010  SS: 0018
 #9 [fffffb3a080b27ba8] vprintk_emit at ffffffff81b19a91
#10 [fffffb3a080b27bf0] printk at ffffffff81b1a320
#11 [fffffb3a080b27c58] lru_list_show at ffffffffc04c1e65 [get_lru_info]
#12 [fffffb3a080b27cd0] iter_lruvec at ffffffffc04c1fa2 [get_lru_info]
#13 [fffffb3a080b27d68] lru_iter_begin at ffffffffc04c23a5 [get_lru_info]
#14 [fffffb3a080b27df0] proc_dump_write at ffffffffc04c2729 [get_lru_info]
#15 [fffffb3a080b27e30] proc_reg_write at ffffffff81d61209
#16 [fffffb3a080b27e48] __vfs_write at ffffffff81ccfca6
#17 [fffffb3a080b27ec8] vfs_write at ffffffff81ccffbd
#18 [fffffb3a080b27ef8] ksys_write at ffffffff81cd025a
#19 [fffffb3a080b27f38] do_syscall_64 at ffffffff81a0432f
#20 [fffffb3a080b27f50] entry_SYSCALL_64_after_hwframe at ffffffff82400088
    RIP: 00007fb476f89484  RSP: 00007ffef1482878  RFLAGS: 00000246
    RAX: ffffffffffffffda  RBX: 0000000000000005  RCX: 00007fb476f89484
    RDX: 0000000000000005  RSI: 000055c6b974a340  RDI: 0000000000000001
    RBP: 000055c6b974a340   R8: 000000000000000a   R9: 00000000ffffffff
    R10: 000000000000000a  R11: 0000000000000246  R12: 00007fb4770525c0
    R13: 0000000000000005  R14: 00007fb4770527c0  R15: 0000000000000005
    ORIG_RAX: 0000000000000001  CS: 0033  SS: 002b
crash> dis -l console_unlock+801
/usr/src/debug/kernel-4.19.90-2206.4.0.0156.u55.fos22.x86_64/linux-4.19.90-2206.4.0.0156.u55.fos22.x86_64/./arch/x86/include/asm/paravirt.h: 789
0xffffffff81b17dd1 <console_unlock+801>:        nopl   0x0(%rax,%rax,1)
crash>
```

console_unlock+801 的实际代码功能是 arch_local_irq_restore()，恢复中断。从流程看来是中断被关了太长时间，并且长时间没有喂狗（watchdog task 进程没有更新 timestamp），导致中断恢复的第一时间，就执行了 watchdog（watchdog timer 检查当前时间与 timestap 差值）。

分析为什么 watchdog 中断长时间没有执行：

观察海光虚拟机的 vnc 界面，发现执行时所有信息都打印到 tty 前台，打印信息量非常大。分析 vmcore 信息，确认当前系统 console 就是 tty

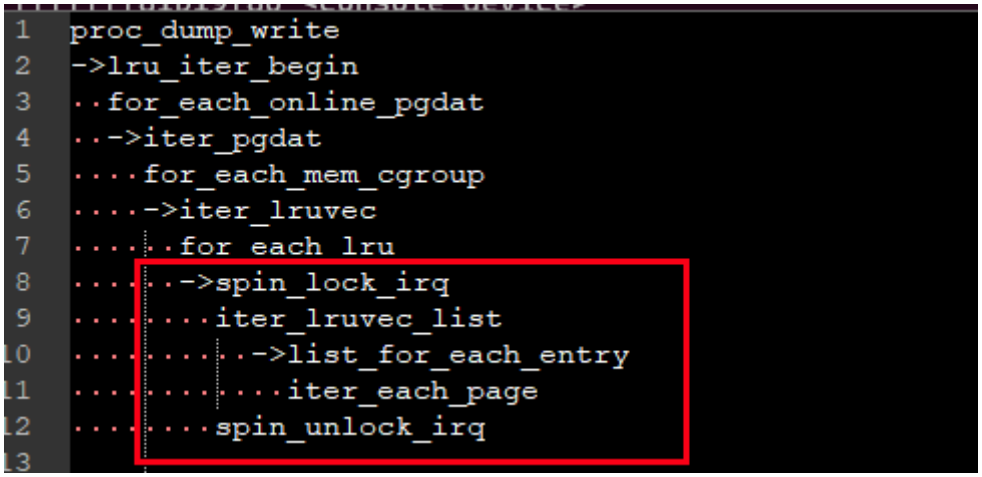
```
crash> p console_device
console_device = $1 =
 {struct tty_driver *(int *)} 0xffffffff81b19f80 <console_device>
crash> sym 0xffffffff81b19f80
ffffffff81b19f80 (T) console_device /usr/src/debug/kernel-4.19.90-2206.4.0.0156.u55.fos22.x
crash>
```

验证：通过调高系统 loglevel，不准许信息打印到前台，发现触发问题的命令很快返回，

系统无 softlockup，无复位。

结论：串口打印太慢导致长时间在刷新串口__log_buff 导致

结合.ko 源码分析，在最内层循环中是关闭中断，并且打印数量很大，导致串口堵塞，在狗叫的时钟周期内 lru 检测线程不能退出循环让 watchdog task 去喂狗，导致出现问题。



三、处理方案

在执行 dump 的入口处调高当前系统的日志打印级别到 1，避免 console 日志打印到 tty。dump 结束之后回复原来的日志等级。

```
change_console_loglevel();  // 调整打印级别
for … …
    … …
restore_console_loglevel();  // 恢复日志级别
```

四、问题回归

回归测试系统正常运行，LRU 信息输出正常。