
Subject: Re: Sv: Re: Infinite loop in `__d_lookup` ?
Posted by [Pavel Emelianov](#) on Thu, 15 May 2008 16:21:49 GMT
[View Forum Message](#) <> [Reply to Message](#)

Jakob Goldbach wrote:

> That would be great. Thanks. There are usually a few days between it gets stuck.

Ok. Happily, I've managed to invent what I need to check first before it's too late here in Moscow ;)

I presume, that the infinite loop is really somewhere near the `__d_lookup`. Please, apply this patch in attach (I made it against 2.6.18-028stab053.5, but should fit OK all the other 028stab053 releases) and check for warnings in `dmesg` ;)

Let's see whether this is really `__d_lookup`.

> /jakob

> - oprindeligt besked -

> Emne: Re: [Users] Infinite loop in `__d_lookup` ?

> Fra: Pavel Emelyanov <xemul@openvz.org>

> Dato: 15-05-2008 12:34

>

> Jakob Goldbach wrote:

>> Hi,

>>

>> I regularly have processes that gets stuck eating all cpu. `SysRq-p` says

>> it is stuck in `__d_lookup+0x10b` as seen in `dmesg` output below.

>

> If you can reproduce this in a reasonable time I can send you

> a debugging patch to find out what's going on there.

>

> Let's try with it?

>

>> I run vanilla 2.6.18 with 028stab053 and the lustre filesystem. I also

>> run lustre on non-openvz kernel without problems, hence this mail to

>> this group.

>>

>> I believe I've found where the problem is, but I'm not a kernel hacker

>> so I don't know what to do about this information.

>>

>> I'd appreciate any hints on what to do next to get this solved.

>>

>> Below is what I could find out.

>>

>> Thanks,

>> Jakob

>>

```

>> gdb find that the process is in the hlist_for_each_entry_rcu loop:
>>
>> (gdb) list *__d_lookup+0x10b
>> 0x12f0 is in __d_lookup (fs/dcache.c:1153).
>> 1148     struct dentry *dentry, *found;
>> 1149
>> 1150     rcu_read_lock();
>> 1151
>> 1152     found = NULL;
>> 1153     hlist_for_each_entry_rcu(dentry, node, head, d_hash) {
>> 1154         struct qstr *qstr;
>> 1155
>> 1156         if (dentry->d_name.hash != hash)
>> 1157             continue;
>>
>> I believe this is the relevant part (0x12f0) of the disassembled object:
>>
>> 12e0:  4d 8b 24 24     mov  (%r12),%r12
>> 12e4:  4d 85 e4        test %r12,%r12
>> 12e7:  74 2c          je   1315 <__d_lookup+0x130>
>> 12e9:  49 8b 04 24     mov  (%r12),%rax
>> 12ed:  0f 18 08       prefetch0 (%rax)
>> 12f0:  49 8d 5c 24 d8  lea 0xffffffffffd8(%r12),
>> %rbx
>> 12f5:  8b 45 cc       mov  0xfffffffffcc(%rbp),
>> %eax
>> 12f8:  39 43 40       cmp  %eax,0x40(%rbx)
>> 12fb:  75 e3         jne 12e0 <__d_lookup+0xfb>
>>
>>
>> Dmesg after sysrq-p:
>>
>>
>>
>> [186124.494329] SysRq: Show Regs
>> [186124.495218] ----- IPI show regs -----
>> [186124.496136] CPU 3, VCPU 0:1
>> [186124.496804] Modules linked in: simfs vznetdev vzethdev vzrst ip_nat
>> vzcpt ip_contrack nfnetlink vzdquota vzmon vzdev xt_length ipt_ttl xt_
>> tcpmss ipt_TCPMSS iptable_mangle xt_multiport xt_limit ipt_tos
>> ipt_REJECT iptable_filter ip_tables x_tables 8021q osc mgc lustre lov
>> lquota mdc
>> ksocklnd ptilrc obdclass lnet lvfs libcfs bonding xfs
>> [186124.503636] Pid: 22699, comm: find Not tainted
>> 2.6.18.8-openvz-028stab053-bnx2-1.6.7b-arpanounce1 #3 028stab053
>> [186124.505535] RIP: 0060:[<ffffff8029b314>] [<ffffff8029b314>]
>> __d_lookup+0x10b/0x142

```

```

>> [186124.507265] RSP: 0068:ffff810073d63bc8 EFLAGS: 00000282
>> [186124.508296] RAX: ffff8101016dc298 RBX: ffff8101016dc270 RCX:
>> 00000000000000013
>> [186124.509768] RDX: 0000000000025ff5 RSI: 00c38320c56a5ff5 RDI:
>> ffff810118b056b0
>> [186124.511480] RBP: ffff810073d63c08 R08: ffff8100ac9e8000 R09:
>> ffff810118b056b0
>> [186124.512963] R10: 0000000000000000 R11: 0000000000000000 R12:
>> ffff8101016dc298
>> [186124.514452] R13: ffff810073d63e38 R14: ffff810118b056b0 R15:
>> ffff810073d63c78
>> [186124.515931] FS: 00002ba786cb56d0(0000) GS:ffff81012a693340(0000)
>> knlGS:0000000000000000
>> [186124.517538] CS: 0060 DS: 0000 ES: 0000 CR0: 0000000080050033
>> [186124.518587] CR2: 0000000000539938 CR3: 0000000073f06000 CR4:
>> 000000000000006e0
>> [186124.520022]
>> [186124.520023] Call Trace:
>> [186124.521245] [<ffffffff8029105d>] do_lookup+0x2c/0x193
>> [186124.522363] [<ffffffff80293122>] __link_path_walk+0xb07/0x10ac
>> [186124.523642] [<ffffffff8029374e>] link_path_walk+0x87/0x140
>> [186124.524818] [<ffffffff80293c76>] do_path_lookup+0x2d3/0x2f8
>> [186124.526000] [<ffffffff802945e2>] __user_walk_fd+0x41/0x62
>> [186124.527156] [<ffffffff8028cecb>] vfs_lstat_fd+0x24/0x5a
>> [186124.528278] [<ffffffff8028cf23>] sys_newlstat+0x22/0x3c
>> [186124.529383] [<ffffffff80209902>] system_call+0x7e/0x83
>> [186124.530362] DWARF2 unwinder stuck at system_call+0x7e/0x83
>> [186124.531460] Leftover inexact backtrace:
>> [186124.532563]
>>
>>
--- ./fs/dcache.c.loopdebug 2008-05-15 20:09:04.000000000 +0400
+++ ./fs/dcache.c 2008-05-15 20:16:19.000000000 +0400
@@ -1128,12 +1128,24 @@ struct dentry * d_lookup(struct dentry *
{
    struct dentry * dentry = NULL;
    unsigned long seq;
+ unsigned long loops = 0;
+ static int once = 1;

    do {
        seq = read_seqbegin(&rename_lock);
        dentry = __d_lookup(parent, name);
        if (dentry)
            break;
+
+ if (loops++ > 200) {
+     printk("%s: Abort on 200 seq-retry iteration\n",

```

```

+ __func__);
+ if (once) {
+ once = 0;
+ dump_stack();
+ }
+ break;
+ }
} while (read_seqretry(&rename_lock, seq));
return dentry;
}
@@ -1146,6 +1158,8 @@ struct dentry * __d_lookup(struct dentry
struct hlist_head *head = d_hash(parent,hash);
struct hlist_node *node;
struct dentry *dentry, *found;
+ unsigned long loops = 0;
+ static int once = 1;

rcu_read_lock();

@@ -1154,9 +1168,9 @@ struct dentry * __d_lookup(struct dentry
struct qstr *qstr;

if (dentry->d_name.hash != hash)
- continue;
+ goto next_nolock;
if (dentry->d_parent != parent)
- continue;
+ goto next_nolock;

spin_lock(&dentry->d_lock);

@@ -1193,6 +1207,16 @@ struct dentry * __d_lookup(struct dentry
break;
next:
spin_unlock(&dentry->d_lock);
+next_nolock:
+ if (loops++ > 5000) {
+ printk("%s: Abort on 5000 loop iteration in a chain\n",
+ __func__);
+ if (once) {
+ once = 0;
+ dump_stack();
+ }
+ break;
+ }
}
rcu_read_unlock();

```
