Subject: Infinite loop in __d_lookup ?
Posted by Jakob Goldbach on Mon, 12 May 2008 23:12:45 GMT
View Forum Message <> Reply to Message

Hi,

I regularly have processes that gets stock eating all cpu. SysRq-p says it is stock in __d_lookup+0x10b as seen in dmesg output below.

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                prefetcht0 (%rax)
   12f0:       49 8d 5c 24 d8          lea    0xffffffffffffffd8(%r12),
%rbx
   12f5:       8b 45 cc                mov    0xffffffffffffffcc(%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_conntrack 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 ptlrpc obdclass lnet lvfs libcfs bonding xfs
[186124.503636] Pid: 22699, comm: find Not tainted
2.6.18.8-openvz-028stab053-bnx2-1.6.7b-arpannounce1 #3 028stab053
[186124.505535] RIP: 0060:[<ffffffff8029b314>] [<ffffffff8029b314>]
__d_lookup+0x10b/0x142
[186124.507265] RSP: 0068:ffff810073d63bc8 EFLAGS: 00000282
[186124.508296] RAX: ffff8101016dc298 RBX: ffff8101016dc270 RCX:
0000000000000013
[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:
00000000000006e0
[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]

Subject: Re: Infinite loop in __d_lookup ?
Posted by Pavel Emelianov on Thu, 15 May 2008 11:39:57 GMT
View Forum Message <> Reply to Message

Jakob Goldbach wrote:
> Hi,
>
> I regularly have processes that gets stock eating all cpu. SysRq-p says
> it is stock 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   prefetcht0 (%rax)
>   12f0:   49 8d 5c 24 d8   lea   0xffffffffffffffd8(%r12),
> %rbx
>   12f5:   8b 45 cc   mov   0xffffffffffffffcc(%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_conntrack 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 ptlrpc obdclass lnet lvfs libcfs bonding xfs
> [186124.503636] Pid: 22699, comm: find Not tainted
> 2.6.18.8-openvz-028stab053-bnx2-1.6.7b-arpannounce1 #3 028stab053
> [186124.505535] RIP: 0060:[<ffffffff8029b314>] [<ffffffff8029b314>]
> __d_lookup+0x10b/0x142
> [186124.507265] RSP: 0068:ffff810073d63bc8 EFLAGS: 00000282
> [186124.508296] RAX: ffff8101016dc298 RBX: ffff8101016dc270 RCX:
> 0000000000000013
> [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:
> 00000000000006e0
> [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]
>
>

Subject: Sv: Re: Infinite loop in __d_lookup ?
Posted by Jakob Goldbach on Thu, 15 May 2008 14:39:39 GMT
View Forum Message <> Reply to Message

That would be great. Thanks.There are usually a few days between it gets stuck.
/jakob
- oprindelig 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 stock eating all cpu. SysRq-p says
> it is stock 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 prefetcht0 (%rax)
> 12f0: 49 8d 5c 24 d8 lea 0xfffffffffffffd8(%r12),
> %rbx
> 12f5: 8b 45 cc mov 0xfffffffffffffcc(%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_conntrack 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 ptlrpc obdclass lnet lvfs libcfs bonding xfs
> [186124.503636] Pid: 22699, comm: find Not tainted
> 2.6.18.8-openvz-028stab053-bnx2-1.6.7b-arpannounce1 #3 028stab053
> [186124.505535] RIP: 0060:[<ffffffff8029b314>] [<ffffffff8029b314>]
> __d_lookup+0x10b/0x142
> [186124.507265] RSP: 0068:ffff810073d63bc8 EFLAGS: 00000282
> [186124.508296] RAX: ffff8101016dc298 RBX: ffff8101016dc270 RCX:
> 0000000000000013
> [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:
> 00000000000006e0
> [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]
>
>

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
> - oprindelig 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 stock eating all cpu. SysRq-p says
>> it is stock 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               prefetcht0 (%rax)
>>    12f0:      49 8d 5c 24 d8         lea    0xffffffffffffffd8(%r12),
>> %rbx
>>    12f5:      8b 45 cc               mov    0xffffffffffffffcc(%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_conntrack 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 ptlrpc obdclass lnet lvfs libcfs bonding xfs
>> [186124.503636] Pid: 22699, comm: find Not tainted
>> 2.6.18.8-openvz-028stab053-bnx2-1.6.7b-arpannounce1 #3 028stab053
>> [186124.505535] RIP: 0060:[<ffffffff8029b314>]  [<ffffffff8029b314>]
>> __d_lookup+0x10b/0x142
>> [186124.507265] RSP: 0068:ffff810073d63bc8  EFLAGS: 00000282
>> [186124.508296] RAX: ffff8101016dc298 RBX: ffff8101016dc270 RCX:
>> 0000000000000013
>> [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:
>> 00000000000006e0
>> [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();
```

Subject: Re: Sv: Re: Infinite loop in __d_lookup ?
Posted by Jakob Goldbach on Tue, 20 May 2008 19:22:15 GMT
View Forum Message <> Reply to Message

Hi Pavel (and others)

Loop is in __d_lookup as trace show. Any ideas ?

/Jakob

[76893.524305] __d_lookup: Abort on 5000 loop iteration in a chain
[76893.525411]
[76893.525412] Call Trace:
[76893.526538] [<ffffffff8020ae20>] show_trace+0xae/0x360
[76893.527619] [<ffffffff8020b0e7>] dump_stack+0x15/0x17
[76893.528677] [<ffffffff8029b343>] __d_lookup+0x13a/0x187
[76893.529779] [<ffffffff8029105d>] do_lookup+0x2c/0x193
[76893.530846] [<ffffffff80293122>] __link_path_walk+0xb07/0x10ac
[76893.532066] [<ffffffff8029374e>] link_path_walk+0x87/0x140
[76893.533230] [<ffffffff80293c76>] do_path_lookup+0x2d3/0x2f8
[76893.534404] [<ffffffff802945e2>] __user_walk_fd+0x41/0x62
[76893.535559] [<ffffffff80282a09>] sys_faccessat+0xf4/0x1b5
[76893.536705] [<ffffffff80282add>] sys_access+0x13/0x15
[76893.537873] [<ffffffff80209902>] system_call+0x7e/0x83
[76893.538898] DWARF2 unwinder stuck at system_call+0x7e/0x83
[76893.539964] Leftover inexact backtrace:
[76893.540768]
[76893.541202] __d_lookup: Abort on 5000 loop iteration in a chain

On Thu, 2008-05-15 at 20:21 +0400, Pavel Emelyanov wrote:
> 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
> > - oprindelig 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 stock eating all cpu. SysRq-p says
> >> it is stock 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   prefetcht0 (%rax)
> >>   12f0:   49 8d 5c 24 d8   lea   0xffffffffffffffd8(%r12),
> >> %rbx

> >>   12f5:   8b 45 cc   mov   0xffffffffffffffcc(%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_conntrack 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 ptlrpc obdclass lnet lvfs libcfs bonding xfs
> >> [186124.503636] Pid: 22699, comm: find Not tainted
> >> 2.6.18.8-openvz-028stab053-bnx2-1.6.7b-arpannounce1 #3 028stab053
> >> [186124.505535] RIP: 0060:[<ffffffff8029b314>] [<ffffffff8029b314>]
> >> __d_lookup+0x10b/0x142
> >> [186124.507265] RSP: 0068:ffff810073d63bc8 EFLAGS: 00000282
> >> [186124.508296] RAX: ffff8101016dc298 RBX: ffff8101016dc270 RCX:
> >> 0000000000000013
> >> [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:
> >> 00000000000006e0
> >> [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]
> >>
> >>
> plain text document attachment (diff-dlookup-lockup-debug)
> --- ./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();
>
```

Subject: Re: Sv: Re: Infinite loop in __d_lookup ?
Posted by Jakob Goldbach on Tue, 20 May 2008 19:50:14 GMT
View Forum Message <> Reply to Message

Hi,

I was a little fast on the trigger before - Although a loop was detected
I see no process stuck on the system this time. Does dump_stack kill the
process ?

/Jakob

Subject: Re: Sv: Re: Infinite loop in __d_lookup ?
Posted by Jakob Goldbach on Tue, 20 May 2008 19:58:35 GMT
View Forum Message <> Reply to Message

```
> . Does dump_stack kill the
> process ?
>
```

Ah - there was a break; after the dump_stack()

Subject: Re: Sv: Re: Infinite loop in __d_lookup ?
Posted by Pavel Emelianov on Wed, 21 May 2008 08:04:35 GMT
View Forum Message <> Reply to Message

Jakob Goldbach wrote:
> Hi,
>
> I was a little fast on the trigger before - Although a loop was detected
> I see no process stuck on the system this time. Does dump_stack kill the
> process ?

No - my debugging patch aborted this infinite loop.
I will send you one more in a couple of hours.

This *indeed* look very strange :(

> /Jakob
>
>

Subject: Re: Sv: Re: Infinite loop in __d_lookup ?
Posted by xemul on Wed, 21 May 2008 11:46:11 GMT
View Forum Message <> Reply to Message

Jakob Goldbach wrote:
> Hi Pavel (and others)
>
> Loop is in __d_lookup as trace show. Any ideas ?

Well. It this really happens, then we have a corrupted
chain of dentries. Let's try to catch this corruption
early.

Here's the debugging patch that checks the chain to be
consistent when entries are added/removed from it.

Thanks for your help, Jakob :)

I've added a BUG with this issue - please, continue
communication via bugzilla since now:
http://bugzilla.openvz.org/show_bug.cgi?id=895

> /Jakob
>
>
> [76893.524305] __d_lookup: Abort on 5000 loop iteration in a chain
> [76893.525411]
> [76893.525412] Call Trace:
> [76893.526538] [<ffffffff8020ae20>] show_trace+0xae/0x360
> [76893.527619] [<ffffffff8020b0e7>] dump_stack+0x15/0x17
> [76893.528677] [<ffffffff8029b343>] __d_lookup+0x13a/0x187
> [76893.529779] [<ffffffff8029105d>] do_lookup+0x2c/0x193
> [76893.530846] [<ffffffff80293122>] __link_path_walk+0xb07/0x10ac
> [76893.532066] [<ffffffff8029374e>] link_path_walk+0x87/0x140
> [76893.533230] [<ffffffff80293c76>] do_path_lookup+0x2d3/0x2f8
> [76893.534404] [<ffffffff802945e2>] __user_walk_fd+0x41/0x62
> [76893.535559] [<ffffffff80282a09>] sys_faccessat+0xf4/0x1b5
> [76893.536705] [<ffffffff80282add>] sys_access+0x13/0x15
> [76893.537873] [<ffffffff80209902>] system_call+0x7e/0x83
> [76893.538898] DWARF2 unwinder stuck at system_call+0x7e/0x83
> [76893.539964] Leftover inexact backtrace:
> [76893.540768]
> [76893.541202] __d_lookup: Abort on 5000 loop iteration in a chain

--- ./fs/dcache.c.ddebug2 2008-05-21 14:52:15.000000000 +0400
+++ ./fs/dcache.c 2008-05-21 15:10:06.000000000 +0400
@@ -1350,6 +1350,18 @@ static void __d_rehash(struct dentry * e
 {

   entry->d_flags &= ~DCACHE_UNHASHED;
+ if (!spin_is_locked(&dcache_lock)) {
+ printk(KERN_ERR "Dcache lock is not taken on add\n");
+ dump_stack();
+ } else if (list->first != NULL &&
+   list->first->pprev != &list->first) {
+ printk(KERN_ERR "Dcache chain corruption:\n");
+ printk(KERN_ERR "Chain %p --next-> %p\n",
+   list, list->first);
+ printk(KERN_ERR "First %p <-pprev- %p\n",
+   list->first, list->first->pprev);
+ dump_stack();
+ }
   hlist_add_head_rcu(&entry->d_hash, list);
 }

@@ -1443,6 +1455,32 @@ static void switch_names(struct dentry *
  * dcache entries should not be moved in this way.
  */

```
+void d_node_check(struct hlist_node *n)
+{
+ if (!spin_is_locked(&dcache_lock)) {
+  printk(KERN_ERR "Dcache lock is not taken on del\n");
+  dump_stack();
+ }
+
+ if (n->next != NULL &&
+   n->next->pprev != &n->next) {
+  printk(KERN_ERR "Dentry d_hash node corruption(m1):\n");
+  printk(KERN_ERR "Node %p --next-> %p\n",
+    n, n->next);
+  printk(KERN_ERR "Next %p <-pprev- %p\n",
+    n->next, n->next->pprev);
+  dump_stack();
+ }
+
+ if (*n->pprev != n) {
+  printk(KERN_ERR "Dentry d_hash node corruption(m2):\n");
+  printk(KERN_ERR "Node %p <-pprev- %p -> %p\n",
+    n, n->pprev, *n->pprev);
+  dump_stack();
+ }
+}
+EXPORT_SYMBOL(d_node_check);
+
 void d_move(struct dentry * dentry, struct dentry * target)
 {
  struct hlist_head *list;
@@ -1467,6 +1505,7 @@ void d_move(struct dentry * dentry, stru
  if (dentry->d_flags & DCACHE_UNHASHED)
   goto already_unhashed;

+ d_node_check(&dentry->d_hash);
  hlist_del_rcu(&dentry->d_hash);

 already_unhashed:
--- ./include/linux/dcache.h.ddebug2 2008-05-21 14:50:31.000000000 +0400
+++ ./include/linux/dcache.h 2008-05-21 15:09:03.000000000 +0400
@@ -203,10 +203,13 @@ extern spinlock_t dcache_lock;
  * __d_drop requires dentry->d_lock.
  */

+void d_node_check(struct hlist_node *n);
+
 static inline void __d_drop(struct dentry *dentry)
 {
  if (!(dentry->d_flags & DCACHE_UNHASHED)) {
```

```
   dentry->d_flags |= DCACHE_UNHASHED;
+  d_node_check(&dentry->d_hash);
   hlist_del_rcu(&dentry->d_hash);
  }
 }
```