Subject: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by KAMEZAWA Hiroyuki on Fri, 14 Dec 2007 07:18:34 GMT
View Forum Message <> Reply to Message

Hi,

While I was testing 2.6.24-rc5-mm1's fair group scheduler (with cgroup),
the system hangs. please confirm. it's reproducible on my box.

My test program is attached.

What happens:
  the system hangs. (panic ?)

Environ:
  ia64/NUMA 8CPU systems. 4 cpus per node.

How to reproduce:
  Compile attached one.
  # gcc -o reg reg.c
  Create group as following
  # mount -t cgroup none /opt/cgroup -o cpu
  # mkdir /opt/cgroup/group_1
  # mkdir /opt/cgroup/group_2

  And run attached program
  # ./reg 8 8

What 'reg' does;
  usage : reg A B C...
  This program forks child process and assign
     A of processes to group_1
     B of processes to group_2
     C of processes to group_3
  kick and waitpid all and repeat.

Thanks,
-Kame

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

File Attachments

1) reg.c, downloaded 376 times

Subject: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by KAMEZAWA Hiroyuki on Fri, 14 Dec 2007 08:17:59 GMT
View Forum Message <> Reply to Message

Tested again, and got NULL access and panic.

This is my guess from stack dump. (raw stack dump is attached below.)
==

```
static struct task_struct *pick_next_task_fair(struct rq *rq)
{
        struct cfs_rq *cfs_rq = &rq->cfs;
        struct sched_entity *se;

        if (unlikely(!cfs_rq->nr_running))
                return NULL;

        do {
                se = pick_next_entity(cfs_rq); <-- se was NULL.
                cfs_rq = group_cfs_rq(se);     <-- se->my_q causes SEGV
        } while (cfs_rq);

        return task_of(se);
}
```

===
Seems first_fair() was NULL in
==

```
static struct sched_entity *pick_next_entity(struct cfs_rq *cfs_rq)
{
        struct sched_entity *se = NULL;

        if (first_fair(cfs_rq)) { <-----------------------------(*)
                se = __pick_next_entity(cfs_rq);
                set_next_entity(cfs_rq, se);
        }

        return se;
}
```

==
from register information.

Thanks,
-Kame

Stack dump is here.
==

```
Pid: 8197, CPU 6, comm:                 reg
psr : 00001210085a2010 ifs : 8000000000000206 ip  : [<a000000100067c01>]    Not tainted
```

ip is at pick_next_task_fair+0x81/0xe0
unat: 0000000000000000 pfs : 0000000000000206 rsc : 0000000000000003
rnat: 0000000000000000 bsps: 0000000000000000 pr : 0000000000556959
ldrs: 0000000000000000 ccv : 0000000000000000 fpsr: 0009804c0270033f
csd : 0000000000000000 ssd : 0000000000000000
b0 : a000000100067c00 b6 : a000000100076a60 b7 : a00000010000ee50
 NaT consumption 2216203124768 [1]^M
Modules linked in: sunrpc binfmt_misc dm_mirror dm_mod fan sg thermal e1000 processor button conta
iner e100 eepro100 mii lpfc mptspi mptscsih mptbase ehci_hcd ohci_hcd uhci_hcd^M
^M
Pid: 8197, CPU 6, comm: reg^M
psr : 00001210085a2010 ifs : 8000000000000206 ip : [<a000000100067c01>]   Not tainted^M
ip is at pick_next_task_fair+0x81/0xe0^M
unat: 0000000000000000 pfs : 0000000000000206 rsc : 0000000000000003^M
rnat: 0000000000000000 bsps: 0000000000000000 pr : 0000000000556959^M
ldrs: 0000000000000000 ccv : 0000000000000000 fpsr: 0009804c0270033f^M
csd : 0000000000000000 ssd : 0000000000000000^M
b0 : a000000100067c00 b6 : a000000100076a60 b7 : a00000010000ee50^M
f6 : 0000000000000000000 f7 : 0000000000000000000^M
f8 : 1003e00000000a0000007 f9 : 1003e00000059499dd2c3^M
f10 : 1003ece02a62ae350c355 f11 : 1003e0000000000000037^M
r1 : a000000100d87a60 r2 : 000000df13538d0b r3 : 0000000000000060^M
r8 : 0000000000000000 r9 : e00001a004034b30 r10 : 0000000000000000^M
r11 : e00001a004034aa8 r12 : e00001a10397fe10 r13 : e00001a103970000^M
r14 : 00000000d594bde3 r15 : e00001a004034ab0 r16 : e00001a004034ab8^M
r17 : e00001a004034ac8 r18 : e00001a004038320 r19 : e00001a10426ff20^M
r20 : 0000000000000000 r21 : 0000000000000000 r22 : 0000000000000001^M
r23 : e00001a004034a91 r24 : e00001a004034a90 r25 : e00001a10426ff10^M
r26 : 0000000000000002 r27 : e00001a0040382f0 r28 : e00001a004038288^M
r29 : a0000001008a5468 r30 : a000000100076a60 r31 : a000000100b726e0^M
^M
Call Trace:^M
 [<a000000100013bc0>] show_stack+0x40/0xa0^M
                        sp=e00001a10397f860 bsp=e00001a103970f18^M
 [<a000000100014840>] show_regs+0x840/0x880^M
                        sp=e00001a10397fa30 bsp=e00001a103970ec0^M
 [<a000000100036fa0>] die+0x1a0/0x2a0^M
                        sp=e00001a10397fa30 bsp=e00001a103970e78^M
 [<a0000001000370f0>] die_if_kernel+0x50/0x80^M
                        sp=e00001a10397fa30 bsp=e00001a103970e48^M
 [<a000000100038260>] ia64_fault+0x1140/0x1260^M
                        sp=e00001a10397fa30 bsp=e00001a103970de8^M
 [<a00000010000ae20>] ia64_leave_kernel+0x0/0x270^M
                        sp=e00001a10397fc40 bsp=e00001a103970de8^M
 [<a000000100067c00>] pick_next_task_fair+0x80/0xe0^M
                        sp=e00001a10397fe10 bsp=e00001a103970db8^M
 [<a0000001006f6a60>] schedule+0x8e0/0x1280^M

```
                       sp=e00001a10397fe10 bsp=e00001a103970d08^M
 [<a000000100074e20>] sys_sched_yield+0xe0/0x100^M
                       sp=e00001a10397fe30 bsp=e00001a103970ca8^M
 [<a00000010000aca0>] ia64_ret_from_syscall+0x0/0x20^M
                       sp=e00001a10397fe30 bsp=e00001a103970ca8^M
 [<a000000000010720>] __kernel_syscall_via_break+0x0/0x20^M
                       sp=e00001a103980000 bsp=e00001a103970ca8^M
```

Disassemble.
==

```
a000000100067b80 <pick_next_task_fair>:
a000000100067b80:       18 10 19 08 80 05       [MMB]       alloc r34=ar.pfs,6,4,0
a000000100067b86:       20 80 83 00 42 00                   adds r2=112,r32
a000000100067b8c:       00 00 00 20                       nop.b 0x0
a000000100067b90:       09 20 81 41 00 21       [MMI]       adds r36=96,r32
a000000100067b96:       00 00 00 02 00 20                   nop.m 0x0
a000000100067b9c:       04 00 c4 00                       mov r33=b0;;
a000000100067ba0:       0b 70 00 04 18 10       [MMI]       ld8 r14=[r2];;
a000000100067ba6:       70 00 38 0c 72 00                   cmp.eq p7,p6=0,r14
a000000100067bac:       00 00 04 00                       nop.i 0x0;;
a000000100067bb0:       10 00 00 00 01 c0       [MIB]       nop.m 0x0
a000000100067bb6:       81 00 00 00 c2 03           (p07) mov r8=r0
a000000100067bbc:       80 00 00 41                 (p07) br.cond.spnt.few a000000100067c30
<pick_next_task_fair+0xb
0>
a000000100067bc0:       09 48 c0 48 00 21       [MMI]       adds r9=48,r36
a000000100067bc6:       00 00 00 02 00 00                   nop.m 0x0
a000000100067bcc:       04 00 00 84                       mov r32=r0;;
a000000100067bd0:       09 00 00 00 01 00       [MMI]       nop.m 0x0
a000000100067bd6:       80 00 24 30 20 00                   ld8 r8=[r9]
a000000100067bdc:       00 00 04 00                       nop.i 0x0;;
a000000100067be0:       03 00 00 00 01 00       [MII]       nop.m 0x0
a000000100067be6:       b0 00 20 14 72 05                   cmp.eq p11,p10=0,r8;;
a000000100067bec:       04 47 fc 8c                 (p10) adds r32=-16,r8;;
a000000100067bf0:       51 29 01 40 00 21       [MIB] (p10) mov r37=r32
a000000100067bf6:       00 00 00 02 00 05                   nop.i 0x0
a000000100067bfc:       58 fe ff 5a               (p10) br.call.dptk.many b0=a000000100067a40
<set_next_entity>;;
a000000100067c00:       0b 18 80 41 00 21       [MMI]       adds r3=96,r32;;
a000000100067c06:       40 02 0c 30 20 00                   ld8 r36=[r3]     <----------panic.
a000000100067c0c:       00 00 04 00                       nop.i 0x0;;
a000000100067c10:       10 00 00 00 01 00       [MIB]       nop.m 0x0
a000000100067c16:       90 00 90 10 72 04                   cmp.eq p9,p8=0,r36
a000000100067c1c:       b0 ff ff 4a               (p08) br.cond.dptk.few a000000100067bc0
<pick_next_task_fair+0x4
```

_____

Containers mailing list

Subject: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by Ingo Molnar on Fri, 14 Dec 2007 09:48:46 GMT
View Forum Message <> Reply to Message

(Cc:-ed other folks as well)

* KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com> wrote:

> Hi,
>
> While I was testing 2.6.24-rc5-mm1's fair group scheduler (with cgroup),
> the system hangs. please confirm. it's reproducible on my box.
>
> My test program is attached.
>
> What happens:
> the system hangs. (panic ?)
>
> Environ:
> ia64/NUMA 8CPU systems. 4 cpus per node.
>
> How to reproduce:
> Compile attached one.
> # gcc -o reg reg.c
> Create group as following
> # mount -t cgroup none /opt/cgroup -o cpu
> # mkdir /opt/cgroup/group_1
> # mkdir /opt/cgroup/group_2
>
> And run attached program
> # ./reg 8 8
>
> What 'reg' does;
> usage : reg A B C...
> This program forks child process and assign
> A of processes to group_1
> B of processes to group_2
> C of processes to group_3
> kick and waitpid all and repeat.
>
> Thanks,
> -Kame

> #include <stdlib.h>

```
> #include <stdio.h>
> #include <strings.h>
> #include <sys/types.h>
> #include <unistd.h>
> #include <sched.h>
> #include <asm/intrinsics.h>
> #include <sys/ipc.h>
> #include <sys/shm.h>
> #include <errno.h>
> #include <sys/times.h>
>
> static char *shared;
> #define MAX_PROCS 32
> #define SHMSIZE (16384)
>
> struct start_stop {
>  int go;
> };
>
> /* Assign PID to a group....
>  * work as # echo PID > /opt/cgroup/group_%d/tasks
>  */
> void assign_to(int pid, int group)
> {
>  FILE *fp;
>  char buf[32];
>
>  memset(buf, 0, sizeof(buf));
>  sprintf(buf,"/opt/cgroup/group_%d/tasks",group);
>  fp = fopen(buf,"w");
>  if (fp == NULL) {
>   perror("fopen");
>   fprintf(stderr, "failed : fopen");
>   exit(0);
>  }
>  fprintf(fp, "%d", pid);
>  fclose(fp);
>  printf("%d to %s\n", pid, buf);
> }
>
> /*
>  * spin wait and go into small loop.
>  * # of loops are counted as score.
>  * This process's utime is recorded in times[id]
>  */
> int worker(int id)
> {
>  struct start_stop *shared_flag;
```

```
>
>  shared_flag = (struct start_stop*)shared;
>  do  {
>   sched_yield();
>   ia64_mf();
>  } while (!shared_flag->go);
> }
>
> /*
>  * If you want to assign..
>  * 2 proces to group 1, 3 procs to group 2 ->#  ./a.out 2 3
>  * 3 proces to group 1, 3 procs to group 2, 3 procs to group 3
>  * ->#  ./a.out 3 3 3
>  * Total 32 procs are supported.
>  */
>
> int main(int argc, char *argv[])
> {
>  int nprocs;
>  int shmid, i;
>  struct start_stop *shared_flag;
>  int pids[MAX_PROCS];
>  int groups[MAX_PROCS];
>
>  memset(pids, 0 , sizeof(pids));
>  memset(groups, 0 , sizeof(groups));
>
> again:
>  for (nprocs = 0, i = 1; i < argc; i++) {
>   int num = atoi(argv[i]);
>   int j;
>   for (j = 0; j < num; j++) {
>    groups[nprocs + j] = i;
>   }
>   nprocs += num;
>  }
>
>  shmid = shmget(IPC_PRIVATE, SHMSIZE, IPC_CREAT | 0666);
>  if (shmid == -1) {
>   perror("shmget");
>   exit(1);
>  }
>
>  shared = shmat(shmid, NULL, 0);
>  shared_flag = (struct start_stop *)shared;
>
>  memset(shared, 0, SHMSIZE);
>  shmctl(shmid, IPC_RMID, 0);
```

```
>
>  for (i = 0; i < nprocs; i++) {
>   int ret;
>   ret = fork();
>   if (ret == 0) {
>    worker(i);
>    exit(0);
>   } else if (ret == -1) {
>    perror("fork");
>    exit(0);
>   }
>   pids[i] = ret;
>  }
>  sleep(1);
>  for (i = 0; i < nprocs; i++)
>   assign_to(pids[i], groups[i]);
>  sleep(1);
>  ia64_mf();
>  shared_flag->go = 1;
>
>  for (i = 0; i < nprocs; i++) {
>   int status;
>   waitpid(pids[i], &status, 0);
>  }
>  goto again;
>
>  return 0;
> }
```

Subject: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by Ingo Molnar on Fri, 14 Dec 2007 09:49:09 GMT
View Forum Message <> Reply to Message

(Cc:-ed other folks as well)

* KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com> wrote:

> Tested again, and got NULL access and panic.
>
> This is my guess from stack dump. (raw stack dump is attached below.)
> ==
>

> static struct task_struct *pick_next_task_fair(struct rq *rq)
> {
> struct cfs_rq *cfs_rq = &rq->cfs;
> struct sched_entity *se;
>
> if (unlikely(!cfs_rq->nr_running))
> return NULL;
>
> do {
> se = pick_next_entity(cfs_rq); <-- se was NULL.
> cfs_rq = group_cfs_rq(se); <-- se->my_q causes SEGV
> } while (cfs_rq);
>
> return task_of(se);
> }
> ===
> Seems first_fair() was NULL in
> ==
> static struct sched_entity *pick_next_entity(struct cfs_rq *cfs_rq)
> {
> struct sched_entity *se = NULL;
>
> if (first_fair(cfs_rq)) { <----------(*)
> se = __pick_next_entity(cfs_rq);
> set_next_entity(cfs_rq, se);
> }
>
> return se;
> }
> ==
> from register information.
>
> Thanks,
> -Kame
>
>
> Stack dump is here.
> ==
> Pid: 8197, CPU 6, comm: reg
> psr : 00001210085a2010 ifs : 8000000000000206 ip : [<a000000100067c01>] Not tainted
> ip is at pick_next_task_fair+0x81/0xe0
> unat: 0000000000000000 pfs : 0000000000000206 rsc : 0000000000000003
> rnat: 0000000000000000 bsps: 0000000000000000 pr : 0000000000556959
> ldrs: 0000000000000000 ccv : 0000000000000000 fpsr: 0009804c0270033f
> csd : 0000000000000000 ssd : 0000000000000000
> b0 : a000000100067c00 b6 : a000000100076a60 b7 : a000000010000e50
> NaT consumption 2216203124768 [1]^M
> Modules linked in: sunrpc binfmt_misc dm_mirror dm_mod fan sg thermal e1000 processor

button conta
> iner e100 eepro100 mii lpfc mptspi mptscsih mptbase ehci_hcd ohci_hcd uhci_hcd^M
> ^M
> Pid: 8197, CPU 6, comm: reg^M
> psr : 00001210085a2010 ifs : 8000000000000206 ip : [<a000000100067c01>]   Not tainted^M
> ip is at pick_next_task_fair+0x81/0xe0^M
> unat: 0000000000000000 pfs : 0000000000000206 rsc : 0000000000000003^M
> rnat: 0000000000000000 bsps: 0000000000000000 pr : 0000000000556959^M
> ldrs: 0000000000000000 ccv : 0000000000000000 fpsr: 0009804c0270033f^M
> csd : 0000000000000000 ssd : 0000000000000000^M
> b0 : a000000100067c00 b6 : a000000100076a60 b7 : a00000010000ee50^M
> f6 : 0000000000000000000 f7 : 0000000000000000000^M
> f8 : 1003e00000000a0000007 f9 : 1003e00000059499dd2c3^M
> f10 : 1003ece02a62ae350c355 f11 : 1003e0000000000000037^M
> r1 : a000000100d87a60 r2 : 000000df13538d0b r3 : 0000000000000060^M
> r8 : 0000000000000000 r9 : e00001a004034b30 r10 : 0000000000000000^M
> r11 : e00001a004034aa8 r12 : e00001a10397fe10 r13 : e00001a103970000^M
> r14 : 00000000d594bde3 r15 : e00001a004034ab0 r16 : e00001a004034ab8^M
> r17 : e00001a004034ac8 r18 : e00001a004038320 r19 : e00001a10426ff20^M
> r20 : 0000000000000000 r21 : 0000000000000000 r22 : 0000000000000001^M
> r23 : e00001a004034a91 r24 : e00001a004034a90 r25 : e00001a10426ff10^M
> r26 : 0000000000000002 r27 : e00001a0040382f0 r28 : e00001a004038288^M
> r29 : a0000001008a5468 r30 : a000000100076a60 r31 : a000000100b726e0^M
> ^M
> Call Trace:^M
> [<a000000100013bc0>] show_stack+0x40/0xa0^M
>   sp=e00001a10397f860 bsp=e00001a103970f18^M
> [<a000000100014840>] show_regs+0x840/0x880^M
>   sp=e00001a10397fa30 bsp=e00001a103970ec0^M
> [<a000000100036fa0>] die+0x1a0/0x2a0^M
>   sp=e00001a10397fa30 bsp=e00001a103970e78^M
> [<a0000001000370f0>] die_if_kernel+0x50/0x80^M
>   sp=e00001a10397fa30 bsp=e00001a103970e48^M
> [<a000000100038260>] ia64_fault+0x1140/0x1260^M
>   sp=e00001a10397fa30 bsp=e00001a103970de8^M
> [<a00000010000ae20>] ia64_leave_kernel+0x0/0x270^M
>   sp=e00001a10397fc40 bsp=e00001a103970de8^M
> [<a000000100067c00>] pick_next_task_fair+0x80/0xe0^M
>   sp=e00001a10397fe10 bsp=e00001a103970db8^M
> [<a0000001006f6a60>] schedule+0x8e0/0x1280^M
>   sp=e00001a10397fe10 bsp=e00001a103970d08^M
> [<a000000100074e20>] sys_sched_yield+0xe0/0x100^M
>   sp=e00001a10397fe30 bsp=e00001a103970ca8^M
> [<a00000010000aca0>] ia64_ret_from_syscall+0x0/0x20^M
>   sp=e00001a10397fe30 bsp=e00001a103970ca8^M
> [<a000000000010720>] __kernel_syscall_via_break+0x0/0x20^M
>   sp=e00001a103980000 bsp=e00001a103970ca8^M
>

> Disassemble.
> ==
> a000000100067b80 <pick_next_task_fair>:
> a000000100067b80: 18 10 19 08 80 05 [MMB] alloc r34=ar.pfs,6,4,0
> a000000100067b86: 20 80 83 00 42 00 adds r2=112,r32
> a000000100067b8c: 00 00 00 20 nop.b 0x0
> a000000100067b90: 09 20 81 41 00 21 [MMI] adds r36=96,r32
> a000000100067b96: 00 00 00 02 00 20 nop.m 0x0
> a000000100067b9c: 04 00 c4 00 mov r33=b0;;
> a000000100067ba0: 0b 70 00 04 18 10 [MMI] ld8 r14=[r2];;
> a000000100067ba6: 70 00 38 0c 72 00 cmp.eq p7,p6=0,r14
> a000000100067bac: 00 00 04 00 nop.i 0x0;;
> a000000100067bb0: 10 00 00 00 01 c0 [MIB] nop.m 0x0
> a000000100067bb6: 81 00 00 00 c2 03 (p07) mov r8=r0
> a000000100067bbc: 80 00 00 41 (p07) br.cond.spnt.few a000000100067c30
<pick_next_task_fair+0xb
> 0>
> a000000100067bc0: 09 48 c0 48 00 21 [MMI] adds r9=48,r36
> a000000100067bc6: 00 00 00 02 00 00 nop.m 0x0
> a000000100067bcc: 04 00 00 84 mov r32=r0;;
> a000000100067bd0: 09 00 00 00 01 00 [MMI] nop.m 0x0
> a000000100067bd6: 80 00 24 30 20 00 ld8 r8=[r9]
> a000000100067bdc: 00 00 04 00 nop.i 0x0;;
> a000000100067be0: 03 00 00 00 01 00 [MII] nop.m 0x0
> a000000100067be6: b0 00 20 14 72 05 cmp.eq p11,p10=0,r8;;
> a000000100067bec: 04 47 fc 8c (p10) adds r32=-16,r8;;
> a000000100067bf0: 51 29 01 40 00 21 [MIB] (p10) mov r37=r32
> a000000100067bf6: 00 00 00 02 00 05 nop.i 0x0
> a000000100067bfc: 58 fe ff 5a (p10) br.call.dptk.many b0=a000000100067a40
<set_next_entity>;;
> a000000100067c00: 0b 18 80 41 00 21 [MMI] adds r3=96,r32;;
> a000000100067c06: 40 02 0c 30 20 00 ld8 r36=[r3] <----------panic.
> a000000100067c0c: 00 00 04 00 nop.i 0x0;;
> a000000100067c10: 10 00 00 00 01 00 [MIB] nop.m 0x0
> a000000100067c16: 90 00 90 10 72 04 cmp.eq p9,p8=0,r36
> a000000100067c1c: b0 ff ff 4a (p08) br.cond.dptk.few a000000100067bc0
<pick_next_task_fair+0x4

_____

Subject: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by KAMEZAWA Hiroyuki on Fri, 14 Dec 2007 10:58:37 GMT
View Forum Message <> Reply to Message

Here is much easier test.

(I'm sorry I'll be absent tomorrow.)

the number of cpus is 8. ia64/NUMA.

The hang occurs when the number of tasks is not smaller than available cpus.
Can be a hint ?

==
[root@rhel51GA testpro]# cat yield.c
#include <sched.h>

int main()
{
	while (1)
		sched_yield();
}

[root@rhel51GA testpro]# cat batch-test.sh
#!/bin/bash -x

mount -t cgroup none /opt/cgroup -o cpu
mkdir /opt/cgroup/group_1
mkdir /opt/cgroup/group_2

./yield &
PIDA=$!
./yield &
PIDB=$!

while true; do
	echo $PIDA > /opt/cgroup/group_1/tasks
	echo $PIDB > /opt/cgroup/group_1/tasks
	echo $PIDA > /opt/cgroup/group_2/tasks;
	echo $PIDB > /opt/cgroup/group_2/tasks
done

[root@rhel51GA testpro]#./batech-test.sh
no hang.

[root@rhel51GA testpro]#taskset 0f ./batech-test.sh
no hang

[root@rhel51GA testpro]#taskset 03 ./batech-test.sh
hang.
Pid: 8132, CPU 0, comm: yield
psr : 00001210085a2010 ifs : 8000000000000206 ip : [<a000000100067c01>]   Not tainted
ip is at pick_next_task_fair+0x81/0xe0

unat: 0000000000000000 pfs : 0000000000000b1d rsc : 0000000000000003
rnat: 0000000000000000 bsps: 0000000000000000 pr : 0000000000566959
ldrs: 0000000000000000 ccv : 0000000000000000 fpsr: 0009804c0270033f
csd : 0000000000000000 ssd : 0000000000000000
b0 : a0000001006f6ac0 b6 : a000000100076a60 b7 : a000000100067b80
f6 : 0000000000000000000 f7 : 0000000000000000000
f8 : 1003e00000000a0000007 f9 : 1003e0000004633b23e65
f10 : 1003ee04f68ea89dfb4c3 f11 : 1003e000000000000002b
r1 : a000000100d87a60 r2 : e0000000011082f0 r3 : 0000000000000060
r8 : 0000000000000000 r9 : e000000001108310 r10 : e000004080032018
r11 : 00000000f86ccc70 r12 : e00000408394fe10 r13 : e000004083940000
r14 : 0000000000000001 r15 : 0000000000000064 r16 : e0000000011089f0
r17 : ffffffffffffffff r18 : e000000001108360 r19 : 0000000000000000
r20 : e00000408003ef10 r21 : 0000000001e9555b r22 : 000000af762794d4
r23 : 00000015e1abc70b r24 : fffffffffffe463 r25 : e00000408003ef10
r26 : 0000000000000002 r27 : e0000000011082f0 r28 : e000000001108288
r29 : a0000001008a5468 r30 : a000000100076a60 r31 : a000000100b726e0

Call Trace:
 [<a000000100013bc0>] show_stack+0x40/0xa0
                        sp=e00000408394f860 bsp=e000004083940f18
 [<a000000100014840>] show_regs+0x840/0x880
                        sp=e00000408394fa30 bsp=e000004083940ec0
 [<a000000100036fa0>] die+0x1a0/0x2a0
                        sp=e00000408394fa30 bsp=e000004083940e78
 [<a0000001000370f0>] die_if_kernel+0x50/0x80
                        sp=e00000408394fa30 bsp=e000004083940e48
 [<a000000100038260>] ia64_fault+0x1140/0x1260
                        sp=e00000408394fa30 bsp=e000004083940de8
 [<a00000010000ae20>] ia64_leave_kernel+0x0/0x270
                        sp=e00000408394fc40 bsp=e000004083940de8
 [<a000000100067c00>] pick_next_task_fair+0x80/0xe0
                        sp=e00000408394fe10 bsp=e000004083940db8
 [<a0000001006f6ac0>] schedule+0x940/0x1280
                        sp=e00000408394fe10 bsp=e000004083940d08
 [<a000000100074e20>] sys_sched_yield+0xe0/0x100
                        sp=e00000408394fe30 bsp=e000004083940ca8
 [<a00000010000aca0>] ia64_ret_from_syscall+0x0/0x20
                        sp=e00000408394fe30 bsp=e000004083940ca8
 [<a000000000010720>] __kernel_syscall_via_break+0x0/0x20
                        sp=e000004083950000 bsp=e000004083940ca8
Thanks,
-Kame

Subject: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by Dhaval Giani on Fri, 14 Dec 2007 11:48:03 GMT
View Forum Message <> Reply to Message

On Fri, Dec 14, 2007 at 07:58:37PM +0900, KAMEZAWA Hiroyuki wrote:
> Here is much easier test.

Thanks for the test! Let me see if I can reproduce it here.

--
regards,
Dhaval

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by Dmitry Adamushko on Fri, 14 Dec 2007 12:47:13 GMT
View Forum Message <> Reply to Message

On 14/12/2007, KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com> wrote:
> Here is much easier test.
> (I'm sorry I'll be absent tomorrow.)
>
> the number of cpus is 8. ia64/NUMA.
>
> The hang occurs when the number of tasks is not smaller than available cpus.
> Can be a hint ?
>
> [...]
>
> [root@rhel51GA testpro]#./batech-test.sh
> no hang.
>
> [root@rhel51GA testpro]#taskset 0f ./batech-test.sh
> no hang
>
> [root@rhel51GA testpro]#taskset 03 ./batech-test.sh
> hang.

have you tried :

[root@rhel51GA testpro]#taskset 01 ./batech-test.sh

hang?

just to be sure SMP does matter here (most likely yes, I guess).

TIA,

>
> Thanks,
> -Kame
>

--
Best regards,
Dmitry Adamushko

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by KAMEZAWA Hiroyuki on Fri, 14 Dec 2007 12:50:44 GMT
View Forum Message <> Reply to Message

-
>have you tried :
>
>[root@rhel51GA testpro]#taskset 01 ./batech-test.sh
>
yes

>hang?
>
no.

>just to be sure SMP does matter here (most likely yes, I guess).
>
maybe. As far as I tested, there was no hang if the number of cpus is 1.

Regards,
-Kame

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by Dhaval Giani on Fri, 14 Dec 2007 14:15:28 GMT

On Fri, Dec 14, 2007 at 01:47:13PM +0100, Dmitry Adamushko wrote:
> On 14/12/2007, KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com> wrote:
> > Here is much easier test.
> > (I'm sorry I'll be absent tomorrow.)
> >
> > the number of cpus is 8. ia64/NUMA.
> >
> > The hang occurs when the number of tasks is not smaller than available cpus.
> > Can be a hint ?
> >
> > [...]
> >
> > [root@rhel51GA testpro]#./batech-test.sh
> > no hang.
> >
> > [root@rhel51GA testpro]#taskset 0f ./batech-test.sh
> > no hang
> >
> > [root@rhel51GA testpro]#taskset 03 ./batech-test.sh
> > hang.
>
> have you tried :
>
> [root@rhel51GA testpro]#taskset 01 ./batech-test.sh
>
> hang?
>
> just to be sure SMP does matter here (most likely yes, I guess).
>

NUMA? I am not able to reproduce it here locally on an x86 8 CPU box.

--
regards,
Dhaval

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: Re: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by KAMEZAWA Hiroyuki on Fri, 14 Dec 2007 14:24:28 GMT

>> just to be sure SMP does matter here (most likely yes, I guess).
>>
>
>NUMA? I am not able to reproduce it here locally on an x86 8 CPU box.
>
yes. I used NUMA. 2 Nodes/4CPU x 2

Hmm..

Thanks,
-Kame

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by Dhaval Giani on Fri, 14 Dec 2007 15:36:07 GMT
View Forum Message <> Reply to Message

On Fri, Dec 14, 2007 at 11:24:28PM +0900, kamezawa.hiroyu@jp.fujitsu.com wrote:
> >> just to be sure SMP does matter here (most likely yes, I guess).
> >>
> >
> >NUMA? I am not able to reproduce it here locally on an x86 8 CPU box.
> >
> yes. I used NUMA. 2 Nodes/4CPU x 2
>

OK, I got hold of an IA64 box, non numa and have managed to reproduce
it.

> Hmm..
>
> Thanks,
> -Kame

--
regards,
Dhaval

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by Dhaval Giani on Fri, 14 Dec 2007 15:38:23 GMT
View Forum Message <> Reply to Message

On Fri, Dec 14, 2007 at 09:06:07PM +0530, Dhaval Giani wrote:
> On Fri, Dec 14, 2007 at 11:24:28PM +0900, kamezawa.hiroyu@jp.fujitsu.com wrote:
> > >> just to be sure SMP does matter here (most likely yes, I guess).
> > >>
> > >
> > >NUMA? I am not able to reproduce it here locally on an x86 8 CPU box.
> > >
> > yes. I used NUMA. 2 Nodes/4CPU x 2
> >
>
> OK, I got hold of an IA64 box, non numa and have managed to reproduce
> it.
>

Actually no, its another bug. Thanks for the program!

reg[3330]: NaT consumption 2216203124768 [1]
Modules linked in: ipv6 button binfmt_misc nls_iso8859_1 loop dm_mod tg3
ext3 jbd fan thermal processor sg mptspi mptscsih mptbase
scsi_transport_spi via82cxxx sd_mod scsi_mod ide_disk ide_core

Pid: 3330, CPU 3, comm: reg
psr : 00001210085a2010 ifs : 8000000000000308 ip : [<a0000001002e0481>]
Not tainted
ip is at rb_erase+0x301/0x7e0
unat: 0000000000000000 pfs : 0000000000000308 rsc : 0000000000000003
rnat: 0000000000000000 bsps: 0000000000000000 pr : a5565666a9556959
ldrs: 0000000000000000 ccv : 0000000000000000 fpsr: 0009804c0270033f
csd : 0000000000000000 ssd : 0000000000000000
b0 : a000000100076290 b6 : a000000100086b20 b7 : a000000100076360
f6 : 1003e0000000000000d34 f7 : 1003e000000000000000a
f8 : 1003e0000000000000000 f9 : 1003e0000000000000152
f10 : 1003e0000000000000004 f11 : 0fff2fffffffff0000000
r1 : a000000100c92030 r2 : e000000244bd0068 r3 : e000000245882000
r8 : e000000245882000 r9 : e000000241e6eda0 r10 : 0000000000000001
r11 : e0000002458f0070 r12 : e0000002458a7d80 r13 : e0000002458a0000
r14 : e000000244bd0060 r15 : e000000244bd0058 r16 : 0000000000000000
r17 : e000000245920d34 r18 : 0000000000000000 r19 : 0000000000000000
r20 : e000000245920c90 r21 : 0000000000000001 r22 : a000000100076360
r23 : a000000100a7f2f8 r24 : a000000100a7f2b0 r25 : e0000002458c0058
r26 : e000000004e05b10 r27 : 0000000000000001 r28 : 0000000000000000
r29 : a000000100a7f2e0 r30 : a000000100a7f2b0 r31 : e000000245920098

Call Trace:
[<a000000100014a80>] show_stack+0x40/0xa0

sp=e0000002458a77d0 bsp=e0000002458a1310
[<a000000100015380>] show_regs+0x840/0x880
sp=e0000002458a79a0 bsp=e0000002458a12b8
[<a0000001000384a0>] die+0x1a0/0x2a0
sp=e0000002458a79a0 bsp=e0000002458a1270
[<a0000001000385f0>] die_if_kernel+0x50/0x80
sp=e0000002458a79a0 bsp=e0000002458a1240
[<a0000001005b1a80>] ia64_fault+0x1180/0x12a0
sp=e0000002458a79a0 bsp=e0000002458a11e0
[<a00000010000b2a0>] ia64_leave_kernel+0x0/0x270
sp=e0000002458a7bb0 bsp=e0000002458a11e0
[<a0000001002e0480>] rb_erase+0x300/0x7e0
sp=e0000002458a7d80 bsp=e0000002458a11a0
[<a000000100076290>] __dequeue_entity+0x70/0xa0
sp=e0000002458a7d80 bsp=e0000002458a1170
[<a000000100076300>] set_next_entity+0x40/0xa0
sp=e0000002458a7d80 bsp=e0000002458a1148
[<a0000001000763a0>] set_curr_task_fair+0x40/0xa0
sp=e0000002458a7d80 bsp=e0000002458a1128
[<a000000100078d90>] sched_move_task+0x2d0/0x340
sp=e0000002458a7d80 bsp=e0000002458a10e8
[<a000000100078e20>] cpu_cgroup_attach+0x20/0x40
sp=e0000002458a7d90 bsp=e0000002458a10b0
[<a0000001000e9370>] attach_task+0x9b0/0xac0
sp=e0000002458a7d90 bsp=e0000002458a1058
[<a0000001000ed4e0>] cgroup_common_file_write+0x340/0x520
sp=e0000002458a7dc0 bsp=e0000002458a1010
[<a0000001000eccd0>] cgroup_file_write+0xf0/0x300
sp=e0000002458a7dd0 bsp=e0000002458a0fc0
[<a00000010017bbd0>] vfs_write+0x1d0/0x320
sp=e0000002458a7e20 bsp=e0000002458a0f70
[<a00000010017c7f0>] sys_write+0x70/0xe0
sp=e0000002458a7e20 bsp=e0000002458a0ef8
[<a00000010000b100>] ia64_ret_from_syscall+0x0/0x20
sp=e0000002458a7e30 bsp=e0000002458a0ef8
[<a000000000010720>] __kernel_syscall_via_break+0x0/0x20
sp=e0000002458a8000 bsp=e0000002458a0ef8

--
regards,
Dhaval

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: Re: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by Dmitry Adamushko on Fri, 14 Dec 2007 16:25:39 GMT
View Forum Message <> Reply to Message

On 14/12/2007, Dhaval Giani <dhaval@linux.vnet.ibm.com> > >
> Actually no, its another bug. Thanks for the program!
>

Humm... this crash is very likely to be caused by the same bug. It
just reveals itself in a different place, but effectivelly the pattern
looks similar. Anyway, the rb-tree gets corrupted... and for both
cases, at the very least the 'current' must be within the tree.
I think, if you repeat your test a number of times, you'll likely get
the very same crash as was reported by Kame.

ia64 does define __ARCH_WANT_UNLOCKED_CTXSW (I checked against
2.6.23.1 that I have at hand)

x86 -- not (it's not reproducible there, right?)

so for ia64 task_running() makes use of 'p->oncpu' to determine
whether a given task is currently running
(as opposed to 'rq->curr == p' otherwise)...

But at first glance, it looks like there shouldn't be situations
leading to some sort of de-synchronization in determining the real
'current'.

Will look at it closer.

>
> --
> regards,
> Dhaval
>

--
Best regards,
Dmitry Adamushko

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: Re: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by Dmitry Adamushko on Fri, 14 Dec 2007 19:51:28 GMT

> [...]
>
> [<a0000001002e0480>] rb_erase+0x300/0x7e0
> [<a000000100076290>] __dequeue_entity+0x70/0xa0
> [<a000000100076300>] set_next_entity+0x40/0xa0
> [<a0000001000763a0>] set_curr_task_fair+0x40/0xa0
> [<a000000100078d90>] sched_move_task+0x2d0/0x340
> [<a000000100078e20>] cpu_cgroup_attach+0x20/0x40
>
> [...]

argh... it's a consequence of the 'current is not kept within the tree" indeed.

When sched_move_task() is called for the 'current' (running on another CPU),
we get the following:

```
...
        running = task_running(rq, tsk);
        on_rq = tsk->se.on_rq;

        if (on_rq) {
                dequeue_task(rq, tsk, 0);
                if (unlikely(running))
                        tsk->sched_class->put_prev_task(rq, tsk);
        }
```

[1] tsk->sched_class->put_prev_task() actually _inserts_ 'tsk' back
into the cfs_rq of its _old_ group :

```
        set_task_cfs_rq(tsk, task_cpu(tsk));
```

[2] now task.se->cfs_rq gets changed

```
        if (on_rq) {
                if (unlikely(running))
                        tsk->sched_class->set_curr_task(rq);
```

[3] and now, tsk->sched_class->set_curr_task(rq) _removes_ the
'current' from the tree... but this tree belongs to the _new_ group
(the task is still within the 'old_group->cfs_rq->rb_tree') ---> oops!

```
                enqueue_task(rq, tsk, 0);
        }
```

Anyway, I have to admit that this problem is a consequence of the
special-case treatment for the 'current' by
'dequeue/enqueue_task()'... it makes the interface less transparent

indeed.

/me thinking on how to get it fixed (e.g. set_task_cfs_rq() might take
care of it) or just get this special-case issue removed (have to check
whether we lose anything in this case)... sigh.

--
Best regards,
Dmitry Adamushko

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by Steven Rostedt on Fri, 14 Dec 2007 21:33:04 GMT
View Forum Message <> Reply to Message

On Fri, 14 Dec 2007, Dmitry Adamushko wrote:

>
> argh... it's a consequence of the 'current is not kept within the tree" indeed.
>

Thanks Dmitry for tracking this down. Although I'm still not convinced we
hit the same bug. But I'm going to go ahead and release 2.6.24-rc5-rt1
anyway. When you have a fix, please CC me and I'll add it to -rt2.

Note: I've added a bunch of logdev (see
http://rostedt.homelinux.com/logdev/README) and I kicked off the hackbench
again. I'll let it run overnight, and if it hits the bug, it will give me
a lot more output to let me know what actually happened.

Thanks,

-- Steve

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: Re: Hang with fair cgroup scheduler (reproducer is attached.)

Posted by Dmitry Adamushko on Sat, 15 Dec 2007 10:22:08 GMT
View Forum Message <> Reply to Message

On 14/12/2007, Steven Rostedt <rostedt@goodmis.org> wrote:
>
> On Fri, 14 Dec 2007, Dmitry Adamushko wrote:
>
> >
> > argh... it's a consequence of the 'current is not kept within the tree" indeed.
> >
>
> Thanks Dmitry for tracking this down.

My analysis was flawed (hmm... me was under control of Belgium beer :-)

The task in not on the runqueue (p->on_rq == 0) at the moment when
put_prev_task_fair() and set_curr_task_fair() get its turn in
sched_move_task()... so dequeue/enqueue_entity() are not triggered,
that's good.

so back to the square #0.

> Thanks,
>
> -- Steve

--
Best regards,
Dmitry Adamushko

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: Re: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by Dhaval Giani on Sat, 15 Dec 2007 10:50:36 GMT
View Forum Message <> Reply to Message

On Sat, Dec 15, 2007 at 11:22:08AM +0100, Dmitry Adamushko wrote:
> On 14/12/2007, Steven Rostedt <rostedt@goodmis.org> wrote:
> >
> > On Fri, 14 Dec 2007, Dmitry Adamushko wrote:
> >
> > >
> > > argh... it's a consequence of the 'current is not kept within the tree" indeed.
> > >
> >
> > Thanks Dmitry for tracking this down.
>
> My analysis was flawed (hmm... me was under control of Belgium beer :-)
>
> The task in not on the runqueue (p->on_rq == 0) at the moment when
> put_prev_task_fair() and set_curr_task_fair() get its turn in
> sched_move_task()... so dequeue/enqueue_entity() are not triggered,
> that's good.
>
> so back to the square #0.
>
> > Thanks,
> >
> > -- Steve
>
> --
> Best regards,
> Dmitry Adamushko

> >
> > Thanks Dmitry for tracking this down.
>
> My analysis was flawed (hmm... me was under control of Belgium beer :-)
>
> The task in not on the runqueue (p->on_rq == 0) at the moment when
> put_prev_task_fair() and set_curr_task_fair() get its turn in
> sched_move_task()... so dequeue/enqueue_entity() are not triggered,
> that's good.
>

Again, I am probably missing something, but if on_rq == 0, then how is
set_curr_task_fair() getting called?

--
regards,
Dhaval

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by Dmitry Adamushko on Sat, 15 Dec 2007 11:15:16 GMT
View Forum Message <> Reply to Message

On 15/12/2007, Dhaval Giani <dhaval@linux.vnet.ibm.com> wrote:
> On Sat, Dec 15, 2007 at 11:22:08AM +0100, Dmitry Adamushko wrote:
> > On 14/12/2007, Steven Rostedt <rostedt@goodmis.org> wrote:
> > >
> > > On Fri, 14 Dec 2007, Dmitry Adamushko wrote:
> > >
> > > >
> > > > argh... it's a consequence of the 'current is not kept within the tree" indeed.
> > > >
> > >
> > > Thanks Dmitry for tracking this down.
> >
> > My analysis was flawed (hmm... me was under control of Belgium beer :-)
> >
> > The task in not on the runqueue (p->on_rq == 0) at the moment when
> > put_prev_task_fair() and set_curr_task_fair() get its turn in
> > sched_move_task()... so dequeue/enqueue_entity() are not triggered,
> > that's good.
> >
>
> Again, I am probably missing something, but if on_rq == 0, then how is

> set_curr_task_fair() getting called?
>

...
	running = task_running(rq, tsk);
	on_rq = tsk->se.on_rq;

// let's say on_rq == 1 , i.e. the task is on the runqueue

	if (on_rq) {
		dequeue_task(rq, tsk, 0);

// now tsk->se.on_rq becomes 0

		if (unlikely(running))
			tsk->sched_class->put_prev_task(rq, tsk);

// put_prev_task() --> put_prev_entity() checks for 'tsk->se.on_rq' to
determine whether __enqueue_entity() must be done ---> and it's 0 in
our case.

[it can be non-zero for the following path : schedule() -->
put_prev_task(..., prev) when deactivate_task(..., prev) was not
previously called in schedule(), i.e. 'prev' was preempted]

tsk->se.on_rq will become 1 only after enqueue_task(). As a result,
tsk->se.on_rq is still 0 when set_curr_task() is executed.

does it make sense now?

> --
> regards,
> Dhaval
>

--
Best regards,
Dmitry Adamushko
_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: Re: Hang with fair cgroup scheduler (reproducer is attached.)

On 15/12/2007, Dmitry Adamushko <dmitry.adamushko@gmail.com> wrote:
>
> My analysis was flawed (hmm... me was under control of Belgium beer :-)
>

ok, I've got another one (just in case... well, this late hour to be
blamed now :-/)

according to Dhaval, we have a crash on ia64 (it's also the arch for
the original report) and it's not reproducible on an otherwise similar
(wrt. # of cpus) x86.

(1) The difference that comes first in mind is that ia64 makes use of
__ARCH_WANT_UNLOCKED_CTXSW

dimm@earth:~/storage/kernel/linux-2.6$ grep -rn
__ARCH_WANT_UNLOCKED_CTXSW include/
include/linux/sched.h:947:#ifdef __ARCH_WANT_UNLOCKED_CTXSW
include/asm-mips/system.h:216:#define __ARCH_WANT_UNLOCKED_CTXSW
include/asm-ia64/system.h:259:#define __ARCH_WANT_UNLOCKED_CTXSW

(2) now, in this case (and for SMP)

task_running() effectively becomes { return p->oncpu; }

(3) consider a case of the context switch between prev --> next on CPU #0

'next' has preempted 'prev'

(4) context_swicth() :

next->oncpu becomes '1' as the result of:

[1] context_switch() --> prepare_task_switch() --> prepare_lock_switch(next) -->
next->oncpu = 1

prev->oncpu becomes '0' as the result of:

[2] context_switch() --> finish_task_switch() -->
finish_lock_switch(prev) --> prev->oncpu = 0

[1] takes place at the very _beginning_ of context_switch() _and_ one

more thing is that rq->lock gets unlocked.

[2] takes place at the very _end_ of context_switch()

Now recall what's task_running() in our case (it's "return task->oncpu")

As a result, between [1] and [2] we have 2 tasks on a single CPU for
which task_running() will return '1' and their runqueue is _unlocked_.

(5) now consider sched_move_task() running on another CPU #1.

due to 'UNLOCKED_CTXSW' it can successfully lock the rq of CPU #0

let's say it's called for 'prev' task (the one being scheduled out on
CPU #0 at this very moment)

as we remember, task_running() returns '1' for it (CPU #0 haven't
reached yet point [2] as described in (4) above)

'prev' is currently on the runqueue (prev->se.on_rq == 1) and within the tree.

what happens is as follows:

- dequeue_task() removes it from the tree ;
- put_prev_task() makes cfs_rq->curr = NULL ;

se == prev.se here... so e.g. __enqueue_entity() is not called for 'prev'

- set_curr_task() --> set_curr_task_fair()

and here things become interesting.

```
static void set_curr_task_fair(struct rq *rq)
{
	struct sched_entity *se = &rq->curr->se;

	for_each_sched_entity(se)
		set_next_entity(cfs_rq_of(se), se);
}
```

so 'se' actually belongs to the 'next' on CPU #0

next->on_rq == 1 (obviously, as dequeue_task() in sched_move_task()
was done for 'prev' !)

and now, set_next_entity() does __dequeue_entity() for 'next' which is
not within the tree !!!

(it's the real 'current' on CPU #0)

that's why the reported oops:

> [<a0000001002e0480>] rb_erase+0x300/0x7e0
> [<a000000100076290>] __dequeue_entity+0x70/0xa0
> [<a000000100076300>] set_next_entity+0x40/0xa0
> [<a0000001000763a0>] set_curr_task_fair+0x40/0xa0
> [<a000000100078d90>] sched_move_task+0x2d0/0x340
> [<a000000100078e20>] cpu_cgroup_attach+0x20/0x40

or maybe there is also a possibility of the rb-tree being corrupted as
a result and having a crash somewhere later (the original report had
another backtrace)

hum... does this analysis make sense to somebody else now?

--
Best regards,
Dmitry Adamushko

Subject: Re: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by Dmitry Adamushko on Sun, 16 Dec 2007 00:00:07 GMT
View Forum Message <> Reply to Message

Dhaval,

so following the analysis in the previous mail... here is a test
patch. Could you please give it a try?

TIA,

(enclosed non white-space broken version)

--- a/kernel/sched.c
+++ b/kernel/sched.c
@@ -7360,7 +7360,7 @@ void sched_move_task(struct task_struct *tsk)

        update_rq_clock(rq);

-       running = task_running(rq, tsk);

```
+       running = (rq->curr == tsk);
        on_rq = tsk->se.on_rq;

        if (on_rq) {
---
```

--
Best regards,
Dmitry Adamushko

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

File Attachments

1) 01-set_task_cfs_rq.patch, downloaded 329 times

Subject: Re: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by Dhaval Giani on Sun, 16 Dec 2007 04:28:21 GMT
View Forum Message <> Reply to Message

On Sun, Dec 16, 2007 at 01:00:07AM +0100, Dmitry Adamushko wrote:
> Dhaval,
>
> so following the analysis in the previous mail... here is a test
> patch. Could you please give it a try?
>

Yep, it works!

Tested-by: Dhaval Giani <dhaval@linux.vnet.ibm.com>

thanks,
--
regards,
Dhaval

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by Ingo Molnar on Sun, 16 Dec 2007 08:55:59 GMT
View Forum Message <> Reply to Message

* Dmitry Adamushko <dmitry.adamushko@gmail.com> wrote:

> --- a/kernel/sched.c
> +++ b/kernel/sched.c
> @@ -7360,7 +7360,7 @@ void sched_move_task(struct task_struct *tsk)
>
> update_rq_clock(rq);
>
> - running = task_running(rq, tsk);
> + running = (rq->curr == tsk);
> on_rq = tsk->se.on_rq;

thanks, i've queued this up (pending more testing).

Btw., you should be able to force the ia64 scheduling by adding this to
the very top of include/linux/sched.h:

#define __ARCH_WANT_UNLOCKED_CTXSW
#define __ARCH_WANT_INTERRUPTS_ON_CTXSW

 Ingo

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by Dmitry Adamushko on Sun, 16 Dec 2007 10:06:22 GMT
View Forum Message <> Reply to Message

On 16/12/2007, Ingo Molnar <mingo@elte.hu> wrote:
>
> * Dmitry Adamushko <dmitry.adamushko@gmail.com> wrote:
>
> > --- a/kernel/sched.c
> > +++ b/kernel/sched.c
> > @@ -7360,7 +7360,7 @@ void sched_move_task(struct task_struct *tsk)
> >
> > update_rq_clock(rq);
> >
> > - running = task_running(rq, tsk);
> > + running = (rq->curr == tsk);
> > on_rq = tsk->se.on_rq;
>
> thanks, i've queued this up (pending more testing).

btw., sched_setscheduler() and rt_mutex_setprio() are also affected

(in general, anything that may call put_prev_task/set_curr_task()
relying task_running()).

Will see, maybe we may come up with smth better than just replacing
task_running() with (rq->curr == tsk) there.

> Btw., you should be able to force the ia64 scheduling by adding this to
> the very top of include/linux/sched.h:
>
> #define __ARCH_WANT_UNLOCKED_CTXSW
> #define __ARCH_WANT_INTERRUPTS_ON_CTXSW

Yeah, with both we even get ARM behavior. Can be a good test indeed.

>
> Ingo
>

--
Best regards,
Dmitry Adamushko

Subject: Re: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by Dmitry Adamushko on Sun, 16 Dec 2007 13:01:17 GMT
View Forum Message <> Reply to Message

Ingo,

what about the following patch instead?

maybe task_is_current() would be a better name though.

Steven,

I guess, there is some analogue of UNLOCKED_CTXSW on -rt
(to reduce contention for rq->lock).
So there can be a race schedule() vs. rt_mutex_setprio() or sched_setscheduler()
for some paths that might explain crashes you have been observing?

I haven't analyzed this case for -rt, so I'm just throwing in the idea in case it can be useful.

From: Dmitry Adamushko <dmitry.adamushko@gmail.com>

sched: introduce task_current()

Some services (e.g. sched_setscheduler(), rt_mutex_setprio() and sched_move_task())
must handle a given task differently in case it's the 'rq->curr' task on its run-queue.
The task_running() interface is not suitable for determining such tasks
for platforms with one of the following options:

#define __ARCH_WANT_UNLOCKED_CTXSW
#define __ARCH_WANT_INTERRUPTS_ON_CTXSW

Due to the fact that it makes use of 'p->oncpu == 1' as a criterion but
such a task is not necessarily 'rq->curr'.

The detailed explanation is available here:
https://lists.linux-foundation.org/pipermail/containers/2007-December/009262.html

Signed-off-by: Dmitry Adamushko <dmitry.adamushko@gmail.com>

---
diff --git a/kernel/sched.c b/kernel/sched.c
index dc6fb24..15d088b 100644
--- a/kernel/sched.c
+++ b/kernel/sched.c
@@ -619,10 +619,15 @@ EXPORT_SYMBOL_GPL(cpu_clock);
 # define finish_arch_switch(prev) do { } while (0)
 #endif

+static inline int task_current(struct rq *rq, struct task_struct *p)
+{
+ return rq->curr == p;
+}
+
 #ifndef __ARCH_WANT_UNLOCKED_CTXSW
 static inline int task_running(struct rq *rq, struct task_struct *p)
 {
- return rq->curr == p;
+ return task_current(rq, p);
 }

 static inline void prepare_lock_switch(struct rq *rq, struct task_struct *next)
@@ -651,7 +656,7 @@ static inline int task_running(struct rq *rq, struct task_struct *p)

```
 #ifdef CONFIG_SMP
  return p->oncpu;
 #else
- return rq->curr == p;
+ return task_current(rq, p);
 #endif
 }

@@ -3340,7 +3345,7 @@ unsigned long long task_sched_runtime(struct task_struct *p)

  rq = task_rq_lock(p, &flags);
  ns = p->se.sum_exec_runtime;
- if (rq->curr == p) {
+ if (task_current(rq, p)) {
   update_rq_clock(rq);
   delta_exec = rq->clock - p->se.exec_start;
   if ((s64)delta_exec > 0)
@@ -4033,7 +4038,7 @@ void rt_mutex_setprio(struct task_struct *p, int prio)

  oldprio = p->prio;
  on_rq = p->se.on_rq;
- running = task_running(rq, p);
+ running = task_current(rq, p);
  if (on_rq) {
   dequeue_task(rq, p, 0);
   if (running)
@@ -4334,7 +4339,7 @@ recheck:
  }
  update_rq_clock(rq);
  on_rq = p->se.on_rq;
- running = task_running(rq, p);
+ running = task_current(rq, p);
  if (on_rq) {
   deactivate_task(rq, p, 0);
   if (running)
@@ -7360,7 +7365,7 @@ void sched_move_task(struct task_struct *tsk)

  update_rq_clock(rq);

- running = task_running(rq, tsk);
+ running = task_current(rq, tsk);
  on_rq = tsk->se.on_rq;

  if (on_rq) {

---
```

_____

Subject: Re: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by Steven Rostedt on Sun, 16 Dec 2007 15:32:54 GMT
View Forum Message <> Reply to Message

On Sun, 16 Dec 2007, Dmitry Adamushko wrote:
> Steven,
>
> I guess, there is some analogue of UNLOCKED_CTXSW on -rt
> (to reduce contention for rq->lock).
> So there can be a race schedule() vs. rt_mutex_setprio() or sched_setscheduler()
> for some paths that might explain crashes you have been observing?
>
> I haven't analyzed this case for -rt, so I'm just throwing in the idea in case it can be useful.

Dmitry,

Thanks! I've been watching this thread on the sidelines with keen
interest. I ran hackbench with my logging over the weekend, and after some
30 hours of running, it finally crashed and gave me a dump of a lot of
interactions in the put_prev_enitity and pick_next_task. I'll analyze this
more on Monday.

-- Steve

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: Re: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by Steven Rostedt on Sun, 16 Dec 2007 23:17:04 GMT
View Forum Message <> Reply to Message

On Sun, 16 Dec 2007, Dmitry Adamushko wrote:

> Steven,
>
> I guess, there is some analogue of UNLOCKED_CTXSW on -rt
> (to reduce contention for rq->lock).
> So there can be a race schedule() vs. rt_mutex_setprio() or sched_setscheduler()
> for some paths that might explain crashes you have been observing?

>
> I haven't analyzed this case for -rt, so I'm just throwing in the idea in case it can be useful.

I haven't fully analyzed this either, but will look much deeper tomorrow.
I wanted to show you this first just to see if you can easily spot what
went wrong.

I used my logdev logging device
http://rostedt.homelinux.com/logdev

The patch that I used to add the logging is here:
http://rostedt.homelinux.com/rt-bug/debug-logdev.patch

To understand this. lfcnprint(...) is just like printk, but it will print
output the following format:

[<timestamp>] cpu:<cpu#> (<current-comm>:<current-pid>) <function>:<line#> <printk-fmt>

The tags of lmark() is just

[<timestamp>] cpu:<cpu#> (<current-comm>:<current-pid>) <function>:<line#>

On context switches, we get

>>>> IN LOGDEV SWITCH <<<< cpu: <cpu#>
CPU=<cpu#> [<timestamp>] <prev>:<prev-pid>(<prios>:<task-state>) -->>
<next>:<next-pid>(<prios>)

The full dmesg with logdump and error backtrace is here:
http://rostedt.homelinux.com/rt-bug/rt-bug.log

Here's a little snippet of where things went wrong.

[94359.652019] cpu:3 (hackbench:1658) pick_next_task_fair:1036 nr_running=1
[94359.652020] cpu:3 (hackbench:1658) pick_next_entity:625 se=ffff810009020800
[94359.652021] cpu:0 (hackbench:1473) put_prev_entity:631
[94359.652022] cpu:3 (hackbench:1658) pick_next_entity:625 se=ffff81003906b5a8
[94359.652022] cpu:2 (softirq-timer/2:32) put_prev_task_rt:283
[94359.652023] cpu:0 (hackbench:1473) put_prev_entity:631
>>>> IN LOGDEV SWITCH <<<< cpu:3
CPU=3 [94359.652023] hackbench:1658(120:120:120:D) -->> hackbench:1586(120:120:120)
>>>> IN LOGDEV SWITCH <<<< cpu:2
CPU=2 [94359.652024] softirq-timer/2:32(49:115:49:D) -->> softirq-rcu/2:39(49:115:49)
>>>> IN LOGDEV SWITCH <<<< cpu:0
CPU=0 [94359.652025] hackbench:1473(120:120:120:R) -->> hackbench:1591(49:120:120)
[94359.652029] cpu:3 (hackbench:1586) put_prev_entity:631
[94359.652030] cpu:2 (softirq-rcu/2:39) move_tasks:2472

[94359.652030] cpu:3 (hackbench:1586) put_prev_entity:631
[94359.652032] cpu:3 (hackbench:1586) pick_next_task_fair:1036 nr_running=1
[94359.652033] cpu:3 (hackbench:1586) pick_next_entity:625 se=ffff810009020800
[94359.652034] cpu:3 (hackbench:1586) pick_next_entity:625 se=ffff810014c7ab18
[94359.652034] cpu:2 (softirq-rcu/2:39) put_prev_task_rt:283
>>>> IN LOGDEV SWITCH <<<< cpu:3
CPU=3 [94359.652035] hackbench:1586(120:120:120:T) -->> hackbench:1623(120:120:120)
[94359.652036] cpu:2 (softirq-rcu/2:39) pick_next_task_fair:1036 nr_running=1
[94359.652038] cpu:2 (softirq-rcu/2:39) pick_next_entity:625 se=0000000000000000

I see that softirq-rcu on cpu 2 started doing a move_tasks, when it got to
the state where nr_running returned 1 and the se from pick_next_entity was
NULL.

This was the run on 2.6.24-rc5-rt1.

I'll look deeper into this on Monday, but if something jumps out at you,
please let me know.

Thanks,

-- Steve

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by KAMEZAWA Hiroyuki on Mon, 17 Dec 2007 01:12:45 GMT
View Forum Message <> Reply to Message

On Sun, 16 Dec 2007 09:58:21 +0530
Dhaval Giani <dhaval@linux.vnet.ibm.com> wrote:

> On Sun, Dec 16, 2007 at 01:00:07AM +0100, Dmitry Adamushko wrote:
> > Dhaval,
> >
> > so following the analysis in the previous mail... here is a test
> > patch. Could you please give it a try?
> >
>
> Yep, it works!
>
> Tested-by: Dhaval Giani <dhaval@linux.vnet.ibm.com>
>

Works for me, too !!

Thanks,
-Kame

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by Dmitry Adamushko on Mon, 17 Dec 2007 10:23:08 GMT
View Forum Message <> Reply to Message

On 17/12/2007, Steven Rostedt <rostedt@goodmis.org> wrote:
>
> Here's a little snippet of where things went wrong.
>
> [94359.652019] cpu:3 (hackbench:1658) pick_next_task_fair:1036 nr_running=1
> [94359.652020] cpu:3 (hackbench:1658) pick_next_entity:625 se=ffff810009020800
> [94359.652021] cpu:0 (hackbench:1473) put_prev_entity:631
> [94359.652022] cpu:3 (hackbench:1658) pick_next_entity:625 se=ffff81003906b5a8
> [94359.652022] cpu:2 (softirq-timer/2:32) put_prev_task_rt:283
> [94359.652023] cpu:0 (hackbench:1473) put_prev_entity:631
> >>>> IN LOGDEV SWITCH <<<< cpu:3
> CPU=3 [94359.652023] hackbench:1658(120:120:120:D) -->> hackbench:1586(120:120:120)
> >>>> IN LOGDEV SWITCH <<<< cpu:2
> CPU=2 [94359.652024] softirq-timer/2:32(49:115:49:D) -->> softirq-rcu/2:39(49:115:49)
> >>>> IN LOGDEV SWITCH <<<< cpu:0
> CPU=0 [94359.652025] hackbench:1473(120:120:120:R) -->> hackbench:1591(49:120:120)
> [94359.652029] cpu:3 (hackbench:1586) put_prev_entity:631
> [94359.652030] cpu:2 (softirq-rcu/2:39) move_tasks:2472
> [94359.652030] cpu:3 (hackbench:1586) put_prev_entity:631
> [94359.652032] cpu:3 (hackbench:1586) pick_next_task_fair:1036 nr_running=1
> [94359.652033] cpu:3 (hackbench:1586) pick_next_entity:625 se=ffff810009020800
> [94359.652034] cpu:3 (hackbench:1586) pick_next_entity:625 se=ffff810014c7ab18
> [94359.652034] cpu:2 (softirq-rcu/2:39) put_prev_task_rt:283
> >>>> IN LOGDEV SWITCH <<<< cpu:3
> CPU=3 [94359.652035] hackbench:1586(120:120:120:T) -->> hackbench:1623(120:120:120)
> [94359.652036] cpu:2 (softirq-rcu/2:39) pick_next_task_fair:1036 nr_running=1
> [94359.652038] cpu:2 (softirq-rcu/2:39) pick_next_entity:625 se=0000000000000000
>
> I see that softirq-rcu on cpu 2 started doing a move_tasks, when it got to
> the state where nr_running returned 1 and the se from pick_next_entity was
> NULL.

move_task() is likely to be run from schedule() --> idle_balance() ,

for our case it means that 'softirq-rcu' (which is a RT task) went to
sleep and it was the last task on this CPU (rq->nr_running == 0 -->
idle_balance() was triggered).

It may be related, maybe not. One 'abnormal' thing (at least, it
occurs only once in this log. Should be checked wheather it happens
when the system works fine) is that a few iterations before the oops
happens we observe the following pattern:

CPU=2 [94359.651930] hackbench:1932(120:120:120:T) -->>
hackbench:1591(120:120:120)

CPU=2 [94359.651980] hackbench:1591(49:120:120:T) -->> swapper:0(140:120:140)

swapper (idle)   --> softirq-timer (RT)
softirq-timer (RT) --> softirq-rcu (RT)
softirq-rcu(RT) --> picks up se == 0 for SCHED_NORMAL upon scheduling
out ---> OOPS

'hackbench' was of SCHED_NORMAL upon scheduling _in_, and it's of RT
type (prio: 49 and schedule() --> put_prev_task_rt()) upon scheduling
out.

Unless you run some modified version of 'hackbench', it doesn't chenge
scheduling classes... so maybe a lifted prio is a consequence of the
resource contention with some RT task ?

This 'hackbench' was the last SCHED_NORMAL task to run on this CPU...
so however this NORMAL -> RT transition happened, it might leave a
sched_fair's runqueue corrupted...

(Will try to look more when time allows).

>
> Thanks,
>
> -- Steve

--
Best regards,
Dmitry Adamushko

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by Ingo Molnar on Mon, 17 Dec 2007 14:45:09 GMT
View Forum Message <> Reply to Message

* KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com> wrote:

> > > so following the analysis in the previous mail... here is a test
> > > patch. Could you please give it a try?
> > >
> >
> > Yep, it works!
> >
> > Tested-by: Dhaval Giani <dhaval@linux.vnet.ibm.com>
> >
> Works for me, too !!

thanks guys, i'll push Dmitry's fix out with the next scheduler git
push.

Ingo

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: Re: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by Steven Rostedt on Mon, 17 Dec 2007 17:58:21 GMT
View Forum Message <> Reply to Message

On Mon, 17 Dec 2007, Dmitry Adamushko wrote:

>
> It may be related, maybe not. One 'abnormal' thing (at least, it
> occurs only once in this log. Should be checked wheather it happens
> when the system works fine) is that a few iterations before the oops
> happens we observe the following pattern:
>
> CPU=2 [94359.651930] hackbench:1932(120:120:120:T) -->>
> hackbench:1591(120:120:120)
>
> CPU=2 [94359.651980] hackbench:1591(49:120:120:T) -->> swapper:0(140:120:140)

Note: the 'T' should be a 'D' because my logdev didn't add the change that
-rt does (adding a 'M' state).

Thanks for noticing. The -rt patch has more priority inheritance
situations than vanilla kernel (sleeping spinlocks or semaphors, and even

the Preempt RCU Boost logic).

>
> swapper (idle)   --> softirq-timer (RT)
> softirq-timer (RT) --> softirq-rcu (RT)
> softirq-rcu(RT) --> picks up se == 0 for SCHED_NORMAL upon scheduling
> out ---> OOPS
>
> 'hackbench' was of SCHED_NORMAL upon scheduling _in_, and it's of RT
> type (prio: 49 and schedule() --> put_prev_task_rt()) upon scheduling
> _out_.
>
> Unless you run some modified version of 'hackbench', it doesn't chenge
> scheduling classes... so maybe a lifted prio is a consequence of the
> resource contention with some RT task ?

Yes. Which means it could be an spinlock, mutex, semaphore or RCU read lock. But since it is in the TASK_UNINTERRUPTIBLE state, I'm willing to bet this is a mutex (or converted spinlock).

>
> This 'hackbench' was the last SCHED_NORMAL task to run on this CPU...
> so however this NORMAL -> RT transition happened, it might leave a
> sched_fair's runqueue corrupted...

Could very well have. The PI uses task_setprio (aka. rt_mutex_setprio) to raise the priority. I'll start looking there.

>
> (Will try to look more when time allows).

Thanks, I'll probably spend the rest of the day on this.

-- Steve

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: Re: Hang with fair cgroup scheduler (reproducer is attached.)
Posted by Dmitry Adamushko on Mon, 17 Dec 2007 22:52:37 GMT
View Forum Message <> Reply to Message

[trimmed the cc' list]

On 17/12/2007, Steven Rostedt <rostedt@goodmis.org> wrote:

>
> On Mon, 17 Dec 2007, Dmitry Adamushko wrote:
>
> >
> > It may be related, maybe not. One 'abnormal' thing (at least, it
> > occurs only once in this log. Should be checked wheather it happens
> > when the system works fine) is that a few iterations before the oops
> > happens we observe the following pattern:
> >
> > CPU=2 [94359.651930] hackbench:1932(120:120:120:T) -->>
> > hackbench:1591(120:120:120)
> >
> > CPU=2 [94359.651980] hackbench:1591(49:120:120:T) -->> swapper:0(140:120:140)
>
> Thanks for noticing. The -rt patch has more priority inheritance
> situations than vanilla kernel (sleeping spinlocks or semaphors, and even
> the Preempt RCU Boost logic).

One more thing is that we don't actually see a point where that
'hackbench' gets its priority lifted.

It was scheduled in as a NORMAL task and scheduled out as a RT one.
i.e. the task got its prio elevated while it was running... a
contention with the task on another CPU?

anyway, i.e. this task must have 'p->se.on_rq == 1' and I'd expect to
see "switched_to_rt" message somewhere in between... hmm?
(check_class_changed() shoud have been called in task_setprio()).

btw., we do see one 'switched_from_rt --> switched_to_fair' case for
another 'hackbench' on CPU #0... according to traces, this one might
get a prio lifted while sleeping (it got scheduled in as a RT task).

>
> -- Steve
>

--
Best regards,
Dmitry Adamushko