

Pavel and all,

I've been profiling the different pidns patchsets to chase the perf bottlenecks in the pidns patchset. As i was not getting accurate profiling results with unixbench, I changed the benchmark to use the nptl perf benchmark ingo used when he introduced the generic pidhash back in 2002.

<http://lwn.net/Articles/10368/>

Compared to unixbench, this is a micro benchmark measuring thread creation and destruction which I think is quite relevant of our different patchsets. unixbench is fine but profiling is not really accurate. too much noise. Any other suggestions ?

On a 2 * Intel(R) Xeon(TM) CPU 2.80GHz with 4 GB of RAM, I ran 8 simultaneous, like ingo did :

```
./perf -s 1000000 -t 1 -r 0 -T --sync-join
```

I did that a few times and also changed the load of the machine to see if values were not too dispersed.

kernels used were :

- * 2.6.22-rc1-mm1
- * <http://lxc.sourceforge.net/patches/2.6.22/2.6.22-rc1-mm1-openvz-pidns1/>
- * <http://lxc.sourceforge.net/patches/2.6.22/2.6.22-rc1-mm1-pidns1/>

findings are :

- * definitely better results for suka's patchset. suka's patchset is also getting better results with unixbench on a 2.6.22-rc1-mm1 but the values are really dispersed. can you confirm ?
- * suka's patchset would benefit from some optimization in `init_upid()` and `dup_struct_pid()`
- * it seems that openvz's pachset has some issue with the struct pid cache. not sure what is the reason. may be you can help pavel.

Cheers,

C.

* results for 2.6.22-rc1-mm1

Runtime: 91.635644842 seconds
Runtime: 91.639834248 seconds
Runtime: 93.615069259 seconds
Runtime: 93.664678865 seconds
Runtime: 95.724542035 seconds
Runtime: 95.763572945 seconds
Runtime: 96.444022314 seconds
Runtime: 97.028016189 seconds

* results for 2.6.22-rc1-mm1-pidns

Runtime: 92.054172217 seconds
Runtime: 93.606016039 seconds
Runtime: 93.624093799 seconds
Runtime: 94.992255782 seconds
Runtime: 95.914365693 seconds
Runtime: 98.080396784 seconds
Runtime: 98.674988254 seconds
Runtime: 98.832674972 seconds

* results for 2.6.22-rc1-mm1-openvz-pidns

Runtime: 92.359771573 seconds
Runtime: 96.517435638 seconds
Runtime: 98.328696048 seconds
Runtime: 100.263042244 seconds
Runtime: 101.003111486 seconds
Runtime: 101.371180205 seconds
Runtime: 102.536653818 seconds
Runtime: 102.671519536 seconds

* diffprofile 2.6.22-rc1-mm1 and 2.6.22-rc1-mm1-pidns

2708	11.8%	check_poison_obj
2461	0.0%	init_upid
2445	2.9%	total
2283	183.7%	kmem_cache_free
383	16.9%	kmem_cache_alloc
365	13.6%	__memset
280	0.0%	dup_struct_pid
279	22.9%	__show_regs
278	21.1%	cache_alloc_debugcheck_after
261	11.3%	get_page_from_freelist
223	0.0%	kref_put
203	3.4%	copy_process

```

197 34.4% do_futex
176 5.6% do_exit
86 22.8% cache_alloc_refill
82 28.2% do_fork
69 18.3% sched_balance_self
68 136.0% __free_pages_ok
59 90.8% bad_range
52 4.3% __down_read
51 13.7% account_user_time
50 7.5% copy_thread
43 28.7% put_files_struct
37 264.3% __free_pages
31 18.9% poison_obj
28 82.4% gs_change
26 16.0% plist_check_prev_next
25 192.3% __put_task_struct
23 26.7% __get_free_pages
23 14.6% __put_user_4
23 230.0% alloc_uid
22 9.0% exit_mm
21 12.9% _raw_spin_unlock
21 7.8% mm_release
21 8.6% plist_check_list
20 20.0% drop_futex_key_refs
20 12.0% __up_read
19 48.7% unqueue_me
19 16.4% do_arch_prctl
18 1800.0% dummy_task_free_security
18 58.1% wake_futex
17 47.2% obj_offset
16 16.7% dbg_userword
15 0.0% kref_get
15 150.0% check_irq_off
15 300.0% __rcu_process_callbacks
14 466.7% __switch_to
14 32.6% prepare_to_copy
14 8.2% get_futex_key
14 16.1% __wake_up
13 65.0% rt_mutex_debug_task_free
12 7.1% obj_size
11 19.3% add_wait_queue
11 275.0% put_pid
11 550.0% profile_task_exit
10 9.0% task_nice
9 100.0% __delay
8 57.1% call_rcu
8 7.8% find_extend_vma
8 266.7% ktime_get

```

```

8  23.5% sys_clone
8  25.0% delayed_put_task_struct
7  26.9% task_rq_lock
7  18.9% _spin_lock_irqsave
6   0.0% quicklist_trim
6 100.0% __up_write
-6 -50.0% module_unload_free
-6 -100.0% nr_running
-7 -43.8% _raw_spin_trylock
-7  -2.8% __alloc_pages
-8 -33.3% sysret_check
-8 -28.6% sysret_careful
-8 -50.0% sysret_signal
-8  -1.9% copy_namespaces
-9 -16.7% memmove
-9 -11.5% __phys_addr
-9  -4.5% copy_semundo
-10 -28.6% rwlock_bug
-10 -27.8% wake_up_new_task
-10 -10.4% sched_clock
-10  -6.2% copy_user_generic_unrolled
-11 -100.0% d_validate
-11 -23.9% monotonic_to_bootbased
-11 -10.6% dummy_task_create
-11  -3.7% futex_wake
-12  -3.9% __might_sleep
-13 -100.0% vsnprintf
-14 -13.0% plist_del
-16 -84.2% sighand_ctor
-17 -20.7% debug_rt_mutex_free_waiter
-17 -42.5% release_thread
-18 -29.5% init_waitqueue_head
-19 -100.0% scnprintf
-21 -12.7% copy_files
-22 -47.8% blocking_notifier_call_chain
-23 -11.8% hash_futex
-24 -18.8% call_rcu_bh
-25 -19.8% mmput
-27 -16.5% down_read
-27 -39.7% audit_alloc
-27 -19.9% stub_clone
-28 -16.3% set_normalized_timespec
-32 -74.4% kfree_debugcheck
-35 -30.2% sys_exit
-40 -63.5% down_read_trylock
-43  -8.6% zone_watermark_ok
-49  -7.7% schedule
-53  -5.4% system_call

```

-54 -47.0% __blocking_notifier_call_chain
 -64 -24.8% getnstimeofday
 -66 -7.0% _raw_spin_lock
 -75 -22.9% ktime_get_ts
 -86 -100.0% snprintf
 -86 -12.8% kernel_thread
 -88 -38.1% plist_add
 -93 -5.4% __memcpy
 -100 -59.9% kmem_flagcheck
 -103 -18.5% acct_collect
 -113 -38.3% dbg_redzone1
 -138 -3.9% schedule_tail
 -162 -12.2% _spin_unlock
 -243 -7.3% thread_return
 -268 -83.5% proc_flush_task
 -289 -100.0% d_lookup
 -357 -100.0% d_hash_and_lookup
 -368 -6.1% release_task
 -642 -99.8% vsnprintf
 -816 -100.0% __d_lookup
 -1529 -100.0% number
 -2431 -100.0% alloc_pid

* diffprofile 2.6.22-rc1-mm1 and 2.6.22-rc1-mm1-openvz-pidns

10046 11.8% total
 6896 554.8% kmem_cache_free
 1580 6.9% check_poison_obj
 1222 0.0% alloc_pidmap
 883 39.0% kmem_cache_alloc
 485 128.6% cache_alloc_refill
 263 8.4% do_exit
 223 40.0% acct_collect
 208 32.3% vsnprintf
 196 14.9% cache_alloc_debugcheck_after
 162 4.5% schedule_tail
 147 25.7% do_futex
 138 276.0% __free_pages_ok
 107 8.8% __down_read
 107 43.7% plist_check_list
 105 6.9% number
 101 61.6% poison_obj
 99 54.4% exit_sem
 73 45.6% copy_user_generic_unrolled
 72 42.1% get_futex_key
 67 24.8% mm_release
 60 6.1% system_call
 59 35.3% __up_read

55 22.4% exit_mm
 54 83.1% bad_range
 54 18.3% dbg_redzone1
 52 371.4% __free_pages
 49 376.9% __put_task_struct
 49 15.3% proc_flush_task
 48 13.4% d_hash_and_lookup
 48 14.0% sys_futex
 47 18.6% plist_check_head
 45 19.7% find_vma
 44 5.4% __d_lookup
 43 50.0% __get_free_pages
 41 205.0% rt_mutex_debug_task_free
 38 7.1% futex_wait
 37 3.9% _raw_spin_lock
 36 1800.0% pgd_dtor
 35 13.6% getnstimeofday
 35 109.4% delayed_put_task_struct
 34 33.0% find_extend_vma
 33 42.3% __phys_addr
 32 19.6% plist_check_prev_next
 32 320.0% alloc_uid
 31 4.9% schedule
 30 19.1% __put_user_4
 29 580.0% __rcu_process_callbacks
 29 39.2% ptregscall_common
 28 82.4% gs_change
 27 31.4% snprintf
 27 75.0% obj_offset
 26 173.3% __inc_zone_state
 23 191.7% module_unload_free
 21 0.6% thread_return
 17 10.4% _raw_spin_unlock
 16 59.3% rff_action
 15 10.0% put_files_struct
 15 375.0% debug_rt_mutex_init
 15 150.0% check_irq_off
 14 350.0% put_pid
 14 16.1% __wake_up
 13 650.0% profile_task_exit
 12 33.3% wake_up_new_task
 10 7.4% stub_clone
 8 800.0% dummy_task_free_security
 8 266.7% tasklet_action
 8 6.9% do_arch_prctl
 7 41.2% dump_line
 7 6.5% plist_del
 7 4.2% kmem_flagcheck

7 36.8% up_write
 6 3.6% obj_size
 6 120.0% bad_page
 -6 -27.3% exit_thread
 -6 -66.7% __delay
 -6 -85.7% futex_requeue
 -6 -54.5% sys_vfork
 -6 -11.8% __spin_lock_init
 -7 -46.7% acct_process
 -7 -11.5% init_waitqueue_head
 -8 -20.5% unqueue_me
 -8 -28.6% sysret_careful
 -8 -4.8% copy_files
 -8 -50.0% sysret_signal
 -11 -31.4% rwlock_bug
 -11 -64.7% futexfs_get_sb
 -13 -21.0% debug_rt_mutex_init_waiter
 -13 -10.2% call_rcu_bh
 -13 -1.9% kernel_thread
 -13 -13.5% sched_clock
 -14 -4.8% d_lookup
 -14 -73.7% sighand_ctor
 -15 -30.0% ret_from_sys_call
 -16 -34.8% blocking_notifier_call_chain
 -17 -8.7% hash_futex
 -18 -41.9% prepare_to_copy
 -18 -17.3% dummy_task_create
 -22 -5.1% copy_namespaces
 -23 -6.2% account_user_time
 -24 -29.3% debug_rt_mutex_free_waiter
 -25 -27.5% dbg_redzone2
 -25 -21.6% sys_exit
 -27 -67.5% sched_fork
 -28 -44.4% down_read_trylock
 -29 -30.2% dbg_userword
 -33 -29.7% task_nice
 -34 -79.1% kfree_debugcheck
 -35 -64.8% memmove
 -43 -26.2% down_read
 -43 -18.6% plist_add
 -46 -1.7% __memset
 -46 -26.7% set_normalized_timespec
 -48 -3.6% _spin_unlock
 -57 -11.4% zone_watermark_ok
 -61 -18.6% ktime_get_ts
 -80 -4.7% __memcpy
 -86 -3.7% get_page_from_freelist
 -87 -23.1% sched_balance_self

-152 -22.7% copy_thread
-383 -6.3% copy_process
-920 -15.2% release_task
-1032 -42.5% alloc_pid
-1045 -85.7% __show_regs

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>

Subject: Re: nptl perf bench and profiling with pidns patchsets

Posted by [dev](#) on Mon, 04 Jun 2007 13:48:52 GMT

[View Forum Message](#) <> [Reply to Message](#)

Cedric,

just a small note.

imho it is not correct to check performance with enabled debug in memory allocator since it can influence cache efficiency much.

In you case looks like you have DEBUG_SLAB enabled.

Pavel will recheck as well what influences on this particular test.

BTW, it is strange... But according to Pavel unixbench results were very reproducible. What was the problem in your case?

Kirill

Cedric Le Goater wrote:

> Pavel and all,

>

> I've been profiling the different pidns patchsets to chase the perf

> bottlenecks in the pidns patchset. As i was not getting accurate

> profiling results with unixbench, I changed the benchmark to use the

> nptl perf benchmark ingo used when he introduced the generic pidhash

> back in 2002.

>

> <http://lwn.net/Articles/10368/>

>

> Compared to unixbench, this is a micro benchmark measuring thread

> creation and destruction which I think is quite relevant of our

> different patchsets. unixbench is fine but profiling is not really

> accurate. too much noise. Any other suggestions ?

>

> On a 2 * Intel(R) Xeon(TM) CPU 2.80GHz with 4 GB of RAM, I ran 8

> simultaneous, like ingo did :

>


```

> ./perf -s 1000000 -t 1 -r 0 -T --sync-join
>
> I did that a few times and also changed the load of the machine
> to see if values were not too dispersed.
>
> kernels used were :
>
> * 2.6.22-rc1-mm1
> * http://lxc.sourceforge.net/patches/2.6.22/2.6.22-rc1-mm1-openvz-pidns1/
> * http://lxc.sourceforge.net/patches/2.6.22/2.6.22-rc1-mm1-pidns1/
>
> findings are :
>
> * definitely better results for suka's patchset. suka's patchset is
> also getting better results with unixbench on a 2.6.22-rc1-mm1 but
> the values are really dispersed. can you confirm ?
> * suka's patchset would benefit from some optimization in init_upid()
> and dup_struct_pid()
> * it seems that openvz's patchset has some issue with the struct pid
> cache. not sure what is the reason. may be you can help pavel.
>
> Cheers,
>
> C.
>
>
> * results for 2.6.22-rc1-mm1
>
> Runtime: 91.635644842 seconds
> Runtime: 91.639834248 seconds
> Runtime: 93.615069259 seconds
> Runtime: 93.664678865 seconds
> Runtime: 95.724542035 seconds
> Runtime: 95.763572945 seconds
> Runtime: 96.444022314 seconds
> Runtime: 97.028016189 seconds
>
> * results for 2.6.22-rc1-mm1-pidns
>
> Runtime: 92.054172217 seconds
> Runtime: 93.606016039 seconds
> Runtime: 93.624093799 seconds
> Runtime: 94.992255782 seconds
> Runtime: 95.914365693 seconds
> Runtime: 98.080396784 seconds
> Runtime: 98.674988254 seconds
> Runtime: 98.832674972 seconds
>

```

```

> * results for 2.6.22-rc1-mm1-openssl-pidns
>
> Runtime: 92.359771573 seconds
> Runtime: 96.517435638 seconds
> Runtime: 98.328696048 seconds
> Runtime: 100.263042244 seconds
> Runtime: 101.003111486 seconds
> Runtime: 101.371180205 seconds
> Runtime: 102.536653818 seconds
> Runtime: 102.671519536 seconds
>
>
> * diffprofile 2.6.22-rc1-mm1 and 2.6.22-rc1-mm1-pidns
>
> 2708 11.8% check_poison_obj
> 2461 0.0% init_upid
> 2445 2.9% total
> 2283 183.7% kmem_cache_free
> 383 16.9% kmem_cache_alloc
> 365 13.6% __memset
> 280 0.0% dup_struct_pid
> 279 22.9% __show_regs
> 278 21.1% cache_alloc_debugcheck_after
> 261 11.3% get_page_from_freelist
> 223 0.0% kref_put
> 203 3.4% copy_process
> 197 34.4% do_futex
> 176 5.6% do_exit
> 86 22.8% cache_alloc_refill
> 82 28.2% do_fork
> 69 18.3% sched_balance_self
> 68 136.0% __free_pages_ok
> 59 90.8% bad_range
> 52 4.3% __down_read
> 51 13.7% account_user_time
> 50 7.5% copy_thread
> 43 28.7% put_files_struct
> 37 264.3% __free_pages
> 31 18.9% poison_obj
> 28 82.4% gs_change
> 26 16.0% plist_check_prev_next
> 25 192.3% __put_task_struct
> 23 26.7% __get_free_pages
> 23 14.6% __put_user_4
> 23 230.0% alloc_uid
> 22 9.0% exit_mm
> 21 12.9% _raw_spin_unlock
> 21 7.8% mm_release

```

```

> 21 8.6% plist_check_list
> 20 20.0% drop_futex_key_refs
> 20 12.0% __up_read
> 19 48.7% unqueue_me
> 19 16.4% do_arch_prctl
> 18 1800.0% dummy_task_free_security
> 18 58.1% wake_futex
> 17 47.2% obj_offset
> 16 16.7% dbg_userword
> 15 0.0% kref_get
> 15 150.0% check_irq_off
> 15 300.0% __rcu_process_callbacks
> 14 466.7% __switch_to
> 14 32.6% prepare_to_copy
> 14 8.2% get_futex_key
> 14 16.1% __wake_up
> 13 65.0% rt_mutex_debug_task_free
> 12 7.1% obj_size
> 11 19.3% add_wait_queue
> 11 275.0% put_pid
> 11 550.0% profile_task_exit
> 10 9.0% task_nice
> 9 100.0% __delay
> 8 57.1% call_rcu
> 8 7.8% find_extend_vma
> 8 266.7% ktime_get
> 8 23.5% sys_clone
> 8 25.0% delayed_put_task_struct
> 7 26.9% task_rq_lock
> 7 18.9% _spin_lock_irqsave
> 6 0.0% quicklist_trim
> 6 100.0% __up_write
> -6 -50.0% module_unload_free
> -6 -100.0% nr_running
> -7 -43.8% _raw_spin_trylock
> -7 -2.8% __alloc_pages
> -8 -33.3% sysret_check
> -8 -28.6% sysret_careful
> -8 -50.0% sysret_signal
> -8 -1.9% copy_namespaces
> -9 -16.7% memmove
> -9 -11.5% __phys_addr
> -9 -4.5% copy_semundo
> -10 -28.6% rwlock_bug
> -10 -27.8% wake_up_new_task
> -10 -10.4% sched_clock
> -10 -6.2% copy_user_generic_unrolled
> -11 -100.0% d_validate

```

```

> -11 -23.9% monotonic_to_bootbased
> -11 -10.6% dummy_task_create
> -11 -3.7% futex_wake
> -12 -3.9% __might_sleep
> -13 -100.0% vsnprintf
> -14 -13.0% plist_del
> -16 -84.2% sighand_ctor
> -17 -20.7% debug_rt_mutex_free_waiter
> -17 -42.5% release_thread
> -18 -29.5% init_waitqueue_head
> -19 -100.0% scnprintf
> -21 -12.7% copy_files
> -22 -47.8% blocking_notifier_call_chain
> -23 -11.8% hash_futex
> -24 -18.8% call_rcu_bh
> -25 -19.8% mmpu
> -27 -16.5% down_read
> -27 -39.7% audit_alloc
> -27 -19.9% stub_clone
> -28 -16.3% set_normalized_timespec
> -32 -74.4% kfree_debugcheck
> -35 -30.2% sys_exit
> -40 -63.5% down_read_trylock
> -43 -8.6% zone_watermark_ok
> -49 -7.7% schedule
> -53 -5.4% system_call
> -54 -47.0% __blocking_notifier_call_chain
> -64 -24.8% getnstimeofday
> -66 -7.0% _raw_spin_lock
> -75 -22.9% ktime_get_ts
> -86 -100.0% snprintf
> -86 -12.8% kernel_thread
> -88 -38.1% plist_add
> -93 -5.4% __memcpy
> -100 -59.9% kmem_flagcheck
> -103 -18.5% acct_collect
> -113 -38.3% dbg_redzone1
> -138 -3.9% schedule_tail
> -162 -12.2% _spin_unlock
> -243 -7.3% thread_return
> -268 -83.5% proc_flush_task
> -289 -100.0% d_lookup
> -357 -100.0% d_hash_and_lookup
> -368 -6.1% release_task
> -642 -99.8% vsnprintf
> -816 -100.0% __d_lookup
> -1529 -100.0% number
> -2431 -100.0% alloc_pid

```

```

>
> * diffprofile 2.6.22-rc1-mm1 and 2.6.22-rc1-mm1-openvz-pidns
>
> 10046  11.8% total
> 6896  554.8% kmem_cache_free
> 1580   6.9% check_poison_obj
> 1222   0.0% alloc_pidmap
> 883   39.0% kmem_cache_alloc
> 485  128.6% cache_alloc_refill
> 263   8.4% do_exit
> 223  40.0% acct_collect
> 208  32.3% vsnprintf
> 196  14.9% cache_alloc_debugcheck_after
> 162   4.5% schedule_tail
> 147  25.7% do_futex
> 138 276.0% __free_pages_ok
> 107   8.8% __down_read
> 107  43.7% plist_check_list
> 105   6.9% number
> 101  61.6% poison_obj
> 99   54.4% exit_sem
> 73   45.6% copy_user_generic_unrolled
> 72   42.1% get_futex_key
> 67   24.8% mm_release
> 60    6.1% system_call
> 59   35.3% __up_read
> 55   22.4% exit_mm
> 54   83.1% bad_range
> 54   18.3% dbg_redzone1
> 52 371.4% __free_pages
> 49 376.9% __put_task_struct
> 49   15.3% proc_flush_task
> 48   13.4% d_hash_and_lookup
> 48   14.0% sys_futex
> 47   18.6% plist_check_head
> 45   19.7% find_vma
> 44    5.4% __d_lookup
> 43  50.0% __get_free_pages
> 41 205.0% rt_mutex_debug_task_free
> 38    7.1% futex_wait
> 37    3.9% _raw_spin_lock
> 36 1800.0% pgd_dtor
> 35   13.6% getnstimeofday
> 35  109.4% delayed_put_task_struct
> 34   33.0% find_extend_vma
> 33   42.3% __phys_addr
> 32   19.6% plist_check_prev_next
> 32  320.0% alloc_uid

```

```

> 31 4.9% schedule
> 30 19.1% __put_user_4
> 29 580.0% __rcu_process_callbacks
> 29 39.2% ptregscall_common
> 28 82.4% gs_change
> 27 31.4% snprintf
> 27 75.0% obj_offset
> 26 173.3% __inc_zone_state
> 23 191.7% module_unload_free
> 21 0.6% thread_return
> 17 10.4% _raw_spin_unlock
> 16 59.3% rff_action
> 15 10.0% put_files_struct
> 15 375.0% debug_rt_mutex_init
> 15 150.0% check_irq_off
> 14 350.0% put_pid
> 14 16.1% __wake_up
> 13 650.0% profile_task_exit
> 12 33.3% wake_up_new_task
> 10 7.4% stub_clone
> 8 800.0% dummy_task_free_security
> 8 266.7% tasklet_action
> 8 6.9% do_arch_prctl
> 7 41.2% dump_line
> 7 6.5% plist_del
> 7 4.2% kmem_flagcheck
> 7 36.8% up_write
> 6 3.6% obj_size
> 6 120.0% bad_page
> -6 -27.3% exit_thread
> -6 -66.7% __delay
> -6 -85.7% futex_requeue
> -6 -54.5% sys_vfork
> -6 -11.8% __spin_lock_init
> -7 -46.7% acct_process
> -7 -11.5% init_waitqueue_head
> -8 -20.5% unqueue_me
> -8 -28.6% sysret_careful
> -8 -4.8% copy_files
> -8 -50.0% sysret_signal
> -11 -31.4% rwlock_bug
> -11 -64.7% futexfs_get_sb
> -13 -21.0% debug_rt_mutex_init_waiter
> -13 -10.2% call_rcu_bh
> -13 -1.9% kernel_thread
> -13 -13.5% sched_clock
> -14 -4.8% d_lookup
> -14 -73.7% sighand_ctor

```

> -15 -30.0% ret_from_sys_call
> -16 -34.8% blocking_notifier_call_chain
> -17 -8.7% hash_futex
> -18 -41.9% prepare_to_copy
> -18 -17.3% dummy_task_create
> -22 -5.1% copy_namespaces
> -23 -6.2% account_user_time
> -24 -29.3% debug_rt_mutex_free_waiter
> -25 -27.5% dbg_redzone2
> -25 -21.6% sys_exit
> -27 -67.5% sched_fork
> -28 -44.4% down_read_trylock
> -29 -30.2% dbg_userword
> -33 -29.7% task_nice
> -34 -79.1% kfree_debugcheck
> -35 -64.8% memmove
> -43 -26.2% down_read
> -43 -18.6% plist_add
> -46 -1.7% __memset
> -46 -26.7% set_normalized_timespec
> -48 -3.6% _spin_unlock
> -57 -11.4% zone_watermark_ok
> -61 -18.6% ktime_get_ts
> -80 -4.7% __memcpy
> -86 -3.7% get_page_from_freelist
> -87 -23.1% sched_balance_self
> -152 -22.7% copy_thread
> -383 -6.3% copy_process
> -920 -15.2% release_task
> -1032 -42.5% alloc_pid
> -1045 -85.7% __show_regs
>
>

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

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

Subject: Re: nptl perf bench and profiling with pidns patchsets
Posted by [Cedric Le Goater](#) on Mon, 04 Jun 2007 14:01:32 GMT
[View Forum Message](#) <> [Reply to Message](#)

Kirill Korotaev wrote:

> Cedric,
>
> just a small note.
> imho it is not correct to check performance with enabled debug in memory allocator
> since it can influence cache efficiency much.
> In you case looks like you have DEBUG_SLAB enabled.

you're right. i'll rerun and resend.

> Pavel will recheck as well what influences on this particular test.
> BTW, it is strange... But according to Pavel unixbench results
> were very reproducible. What was the problem in your case?

the results were also very reproducible but the profiling was too noisy.
we also changed the kernel. the previous pidns patchset was on a 2.6.21-mm2
and we ported it on a 2.6.22-rc1-mm1.

but let me remove some debugging options,

thanks,

C.

> Kirill
>
> Cedric Le Goater wrote:
>> Pavel and all,
>>
>> I've been profiling the different pidns patchsets to chase the perf
>> bottlenecks in the pidns patchset. As i was not getting accurate
>> profiling results with unixbench, I changed the benchmark to use the
>> nptl perf benchmark ingo used when he introduced the generic pidhash
>> back in 2002.
>>
>> <http://lwn.net/Articles/10368/>
>>
>> Compared to unixbench, this is a micro benchmark measuring thread
>> creation and destruction which I think is quite relevant of our
>> different patchsets. unixbench is fine but profiling is not really
>> accurate. too much noise. Any other suggestions ?
>>
>> On a 2 * Intel(R) Xeon(TM) CPU 2.80GHz with 4 GB of RAM, I ran 8
>> simultaneous, like ingo did :
>>
>> ./perf -s 1000000 -t 1 -r 0 -T --sync-join
>>
>> I did that a few times and also changed the load of the machine


```

>> to see if values were not too dispersed.
>>
>> kernels used were :
>>
>> * 2.6.22-rc1-mm1
>> * http://lxc.sourceforge.net/patches/2.6.22/2.6.22-rc1-mm1-openvz-pidns1/
>> * http://lxc.sourceforge.net/patches/2.6.22/2.6.22-rc1-mm1-pidns1/
>>
>> findings are :
>>
>> * definitely better results for suka's patchset. suka's patchset is
>> also getting better results with unixbench on a 2.6.22-rc1-mm1 but
>> the values are really dispersed. can you confirm ?
>> * suka's patchset would benefit from some optimization in init_upid()
>> and dup_struct_pid()
>> * it seems that openvz's patchset has some issue with the struct pid
>> cache. not sure what is the reason. may be you can help pavel.
>>
>> Cheers,
>>
>> C.
>>
>>
>> * results for 2.6.22-rc1-mm1
>>
>> Runtime: 91.635644842 seconds
>> Runtime: 91.639834248 seconds
>> Runtime: 93.615069259 seconds
>> Runtime: 93.664678865 seconds
>> Runtime: 95.724542035 seconds
>> Runtime: 95.763572945 seconds
>> Runtime: 96.444022314 seconds
>> Runtime: 97.028016189 seconds
>>
>> * results for 2.6.22-rc1-mm1-pidns
>>
>> Runtime: 92.054172217 seconds
>> Runtime: 93.606016039 seconds
>> Runtime: 93.624093799 seconds
>> Runtime: 94.992255782 seconds
>> Runtime: 95.914365693 seconds
>> Runtime: 98.080396784 seconds
>> Runtime: 98.674988254 seconds
>> Runtime: 98.832674972 seconds
>>
>> * results for 2.6.22-rc1-mm1-openvz-pidns
>>
>> Runtime: 92.359771573 seconds

```

```

>> Runtime: 96.517435638 seconds
>> Runtime: 98.328696048 seconds
>> Runtime: 100.263042244 seconds
>> Runtime: 101.003111486 seconds
>> Runtime: 101.371180205 seconds
>> Runtime: 102.536653818 seconds
>> Runtime: 102.671519536 seconds
>>
>>
>> * diffprofile 2.6.22-rc1-mm1 and 2.6.22-rc1-mm1-pidns
>>
>> 2708 11.8% check_poison_obj
>> 2461 0.0% init_upid
>> 2445 2.9% total
>> 2283 183.7% kmem_cache_free
>> 383 16.9% kmem_cache_alloc
>> 365 13.6% __memset
>> 280 0.0% dup_struct_pid
>> 279 22.9% __show_regs
>> 278 21.1% cache_alloc_debugcheck_after
>> 261 11.3% get_page_from_freelist
>> 223 0.0% kref_put
>> 203 3.4% copy_process
>> 197 34.4% do_futex
>> 176 5.6% do_exit
>> 86 22.8% cache_alloc_refill
>> 82 28.2% do_fork
>> 69 18.3% sched_balance_self
>> 68 136.0% __free_pages_ok
>> 59 90.8% bad_range
>> 52 4.3% __down_read
>> 51 13.7% account_user_time
>> 50 7.5% copy_thread
>> 43 28.7% put_files_struct
>> 37 264.3% __free_pages
>> 31 18.9% poison_obj
>> 28 82.4% gs_change
>> 26 16.0% plist_check_prev_next
>> 25 192.3% __put_task_struct
>> 23 26.7% __get_free_pages
>> 23 14.6% __put_user_4
>> 23 230.0% alloc_uid
>> 22 9.0% exit_mm
>> 21 12.9% _raw_spin_unlock
>> 21 7.8% mm_release
>> 21 8.6% plist_check_list
>> 20 20.0% drop_futex_key_refs
>> 20 12.0% __up_read

```

```

>> 19 48.7% unqueue_me
>> 19 16.4% do_arch_prctl
>> 18 1800.0% dummy_task_free_security
>> 18 58.1% wake_futex
>> 17 47.2% obj_offset
>> 16 16.7% dbg_userword
>> 15 0.0% kref_get
>> 15 150.0% check_irq_off
>> 15 300.0% __rcu_process_callbacks
>> 14 466.7% __switch_to
>> 14 32.6% prepare_to_copy
>> 14 8.2% get_futex_key
>> 14 16.1% __wake_up
>> 13 65.0% rt_mutex_debug_task_free
>> 12 7.1% obj_size
>> 11 19.3% add_wait_queue
>> 11 275.0% put_pid
>> 11 550.0% profile_task_exit
>> 10 9.0% task_nice
>> 9 100.0% __delay
>> 8 57.1% call_rcu
>> 8 7.8% find_extend_vma
>> 8 266.7% ktime_get
>> 8 23.5% sys_clone
>> 8 25.0% delayed_put_task_struct
>> 7 26.9% task_rq_lock
>> 7 18.9% _spin_lock_irqsave
>> 6 0.0% quicklist_trim
>> 6 100.0% __up_write
>> -6 -50.0% module_unload_free
>> -6 -100.0% nr_running
>> -7 -43.8% _raw_spin_trylock
>> -7 -2.8% __alloc_pages
>> -8 -33.3% sysret_check
>> -8 -28.6% sysret_careful
>> -8 -50.0% sysret_signal
>> -8 -1.9% copy_namespaces
>> -9 -16.7% memmove
>> -9 -11.5% __phys_addr
>> -9 -4.5% copy_semundo
>> -10 -28.6% rwlock_bug
>> -10 -27.8% wake_up_new_task
>> -10 -10.4% sched_clock
>> -10 -6.2% copy_user_generic_unrolled
>> -11 -100.0% d_validate
>> -11 -23.9% monotonic_to_bootbased
>> -11 -10.6% dummy_task_create
>> -11 -3.7% futex_wake

```

```

>> -12 -3.9% __might_sleep
>> -13 -100.0% vsnprintf
>> -14 -13.0% plist_del
>> -16 -84.2% sighand_ctor
>> -17 -20.7% debug_rt_mutex_free_waiter
>> -17 -42.5% release_thread
>> -18 -29.5% init_waitqueue_head
>> -19 -100.0% scnprintf
>> -21 -12.7% copy_files
>> -22 -47.8% blocking_notifier_call_chain
>> -23 -11.8% hash_futex
>> -24 -18.8% call_rcu_bh
>> -25 -19.8% mmpu
>> -27 -16.5% down_read
>> -27 -39.7% audit_alloc
>> -27 -19.9% stub_clone
>> -28 -16.3% set_normalized_timespec
>> -32 -74.4% kfree_debugcheck
>> -35 -30.2% sys_exit
>> -40 -63.5% down_read_trylock
>> -43 -8.6% zone_watermark_ok
>> -49 -7.7% schedule
>> -53 -5.4% system_call
>> -54 -47.0% __blocking_notifier_call_chain
>> -64 -24.8% getnstimeofday
>> -66 -7.0% _raw_spin_lock
>> -75 -22.9% ktime_get_ts
>> -86 -100.0% snprintf
>> -86 -12.8% kernel_thread
>> -88 -38.1% plist_add
>> -93 -5.4% __memcpy
>> -100 -59.9% kmem_flagcheck
>> -103 -18.5% acct_collect
>> -113 -38.3% dbg_redzone1
>> -138 -3.9% schedule_tail
>> -162 -12.2% _spin_unlock
>> -243 -7.3% thread_return
>> -268 -83.5% proc_flush_task
>> -289 -100.0% d_lookup
>> -357 -100.0% d_hash_and_lookup
>> -368 -6.1% release_task
>> -642 -99.8% vsnprintf
>> -816 -100.0% __d_lookup
>> -1529 -100.0% number
>> -2431 -100.0% alloc_pid
>>
>> * diffprofile 2.6.22-rc1-mm1 and 2.6.22-rc1-mm1-openvz-pidns
>>

```

```

>> 10046 11.8% total
>> 6896 554.8% kmem_cache_free
>> 1580 6.9% check_poison_obj
>> 1222 0.0% alloc_pidmap
>> 883 39.0% kmem_cache_alloc
>> 485 128.6% cache_alloc_refill
>> 263 8.4% do_exit
>> 223 40.0% acct_collect
>> 208 32.3% vsnprintf
>> 196 14.9% cache_alloc_debugcheck_after
>> 162 4.5% schedule_tail
>> 147 25.7% do_futex
>> 138 276.0% __free_pages_ok
>> 107 8.8% __down_read
>> 107 43.7% plist_check_list
>> 105 6.9% number
>> 101 61.6% poison_obj
>> 99 54.4% exit_sem
>> 73 45.6% copy_user_generic_unrolled
>> 72 42.1% get_futex_key
>> 67 24.8% mm_release
>> 60 6.1% system_call
>> 59 35.3% __up_read
>> 55 22.4% exit_mm
>> 54 83.1% bad_range
>> 54 18.3% dbg_redzone1
>> 52 371.4% __free_pages
>> 49 376.9% __put_task_struct
>> 49 15.3% proc_flush_task
>> 48 13.4% d_hash_and_lookup
>> 48 14.0% sys_futex
>> 47 18.6% plist_check_head
>> 45 19.7% find_vma
>> 44 5.4% __d_lookup
>> 43 50.0% __get_free_pages
>> 41 205.0% rt_mutex_debug_task_free
>> 38 7.1% futex_wait
>> 37 3.9% _raw_spin_lock
>> 36 1800.0% pgd_dtor
>> 35 13.6% getnstimeofday
>> 35 109.4% delayed_put_task_struct
>> 34 33.0% find_extend_vma
>> 33 42.3% __phys_addr
>> 32 19.6% plist_check_prev_next
>> 32 320.0% alloc_uid
>> 31 4.9% schedule
>> 30 19.1% __put_user_4
>> 29 580.0% __rcu_process_callbacks

```

```

>> 29 39.2% ptregscall_common
>> 28 82.4% gs_change
>> 27 31.4% snprintf
>> 27 75.0% obj_offset
>> 26 173.3% __inc_zone_state
>> 23 191.7% module_unload_free
>> 21 0.6% thread_return
>> 17 10.4% _raw_spin_unlock
>> 16 59.3% rff_action
>> 15 10.0% put_files_struct
>> 15 375.0% debug_rt_mutex_init
>> 15 150.0% check_irq_off
>> 14 350.0% put_pid
>> 14 16.1% __wake_up
>> 13 650.0% profile_task_exit
>> 12 33.3% wake_up_new_task
>> 10 7.4% stub_clone
>> 8 800.0% dummy_task_free_security
>> 8 266.7% tasklet_action
>> 8 6.9% do_arch_prctl
>> 7 41.2% dump_line
>> 7 6.5% plist_del
>> 7 4.2% kmem_flagcheck
>> 7 36.8% up_write
>> 6 3.6% obj_size
>> 6 120.0% bad_page
>> -6 -27.3% exit_thread
>> -6 -66.7% __delay
>> -6 -85.7% futex_requeue
>> -6 -54.5% sys_vfork
>> -6 -11.8% __spin_lock_init
>> -7 -46.7% acct_process
>> -7 -11.5% init_waitqueue_head
>> -8 -20.5% unqueue_me
>> -8 -28.6% sysret_careful
>> -8 -4.8% copy_files
>> -8 -50.0% sysret_signal
>> -11 -31.4% rwlock_bug
>> -11 -64.7% futexfs_get_sb
>> -13 -21.0% debug_rt_mutex_init_waiter
>> -13 -10.2% call_rcu_bh
>> -13 -1.9% kernel_thread
>> -13 -13.5% sched_clock
>> -14 -4.8% d_lookup
>> -14 -73.7% sighand_ctor
>> -15 -30.0% ret_from_sys_call
>> -16 -34.8% blocking_notifier_call_chain
>> -17 -8.7% hash_futex

```

```
>> -18 -41.9% prepare_to_copy
>> -18 -17.3% dummy_task_create
>> -22 -5.1% copy_namespaces
>> -23 -6.2% account_user_time
>> -24 -29.3% debug_rt_mutex_free_waiter
>> -25 -27.5% dbg_redzone2
>> -25 -21.6% sys_exit
>> -27 -67.5% sched_fork
>> -28 -44.4% down_read_trylock
>> -29 -30.2% dbg_userword
>> -33 -29.7% task_nice
>> -34 -79.1% kfree_debugcheck
>> -35 -64.8% memmove
>> -43 -26.2% down_read
>> -43 -18.6% plist_add
>> -46 -1.7% __memset
>> -46 -26.7% set_normalized_timespec
>> -48 -3.6% _spin_unlock
>> -57 -11.4% zone_watermark_ok
>> -61 -18.6% ktime_get_ts
>> -80 -4.7% __memcpy
>> -86 -3.7% get_page_from_freelist
>> -87 -23.1% sched_balance_self
>> -152 -22.7% copy_thread
>> -383 -6.3% copy_process
>> -920 -15.2% release_task
>> -1032 -42.5% alloc_pid
>> -1045 -85.7% __show_regs
>>
>>
>> Containers mailing list
>> Containers@lists.linux-foundation.org
>> https://lists.linux-foundation.org/mailman/listinfo/containers
>>
>
>
```

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

Subject: Re: nptl perf bench and profiling with pidns patchsets
Posted by [dev](#) on Mon, 04 Jun 2007 14:54:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

> the results were also very reproducible but the profiling was too noisy.

> we also changed the kernel. the previous pidns patchset was on a 2.6.21-mm2
> and we ported it on a 2.6.22-rc1-mm1.

If reproducible, then were they the same as Pavel posted?

> but let me remove some debugging options,

sure.

Kirill

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

Subject: Re: nptl perf bench and profiling with pidns patchsets
Posted by [Cedric Le Goater](#) on Mon, 04 Jun 2007 16:06:56 GMT
[View Forum Message](#) <> [Reply to Message](#)

Kirill Korotaev wrote:

>> the results were also very reproducible but the profiling was too noisy.
>> we also changed the kernel. the previous pidns patchset was on a 2.6.21-mm2
>> and we ported it on a 2.6.22-rc1-mm1.
>
> If reproducible, then were they the same as Pavel posted?
>
>> but let me remove some debugging options,
>
> sure.

for info, I just noticed that one of the host running the bench was
using acpi as a clocksource and not tsc bc it was considered fuzzy.

All my apologies.

C.

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

Subject: Re: nptl perf bench and profiling with pidns patchsets
Posted by [Cedric Le Goater](#) on Tue, 05 Jun 2007 21:19:03 GMT
[View Forum Message](#) <> [Reply to Message](#)

Cedric Le Goater wrote:

> Kirill Korotaev wrote:

>>> the results were also very reproducible but the profiling was too noisy.

>>> we also changed the kernel. the previous pidns patchset was on a 2.6.21-mm2

>>> and we ported it on a 2.6.22-rc1-mm1.

>> If reproducible, then were they the same as Pavel posted?

hmm, i don't think I have answered that question clearly.

No, i didn't get the same results (with a working TSC), even with unixbench, and that's why I switched to profiling with ntpl perf. Because the difference between these patchsets is so little, I'm hoping that there might be one them which could be improved to make a real difference.

Right now, i'm getting better results with suka's by a magnitude of 1 or 2%, with unixbench and with ntpl perf. but that does not mean anything because standard deviation is high and there might be scenarii where pavel's patchset is behaving better much better.

So i'll continue studying these pathsets, and run some more tests under an unshared pid namespace. If you have any suggestion for improvements, please propose. Pavel, could I try your multilevel patchset ?

pavel's proposal is very similar to what we've started talking about in 2005 and it fits our requirements. I'm really in favor of finishing this pid namespace :)

thanks,

C.

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>

Subject: Re: ntpl perf bench and profiling with pidns patchsets
Posted by [Pavel Emelianov](#) on Sat, 09 Jun 2007 08:10:25 GMT
[View Forum Message](#) <> [Reply to Message](#)

Cedric Le Goater wrote:

> Pavel and all,

[snip]

> findings are :
>
> * definitely better results for suka's patchset. suka's patchset is
> also getting better results with unixbench on a 2.6.22-rc1-mm1 but
> the values are really dispersed. can you confirm ?
> * suka's patchset would benefit from some optimization in init_upid()
> and dup_struct_pid()

We have found the reason why Suka's patches showed better performance. Some time ago I sent a letter saying that `proc_flush_task()` actually never worked with his patches - that's the main problem. After removing this call from my patches the results turned to those similar to my.

I'd also like to note that broken-out set of patches is not git bisect safe at all. The very first patch of his own OOPSes the node. Some subsequent patches contain misprints that break the compilation, etc.

So I ask you again - let us prepare our patches again and compare the performance one more time.

Thanks,
Pavel

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

Subject: Re: nptl perf bench and profiling with pidns patchsets
Posted by [Alexey Dobriyan](#) on Sat, 09 Jun 2007 08:33:53 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Sat, Jun 09, 2007 at 12:10:25PM +0400, Pavel Emelianov wrote:

> > * definitely better results for suka's patchset. suka's patchset is
> > also getting better results with unixbench on a 2.6.22-rc1-mm1 but
> > the values are really dispersed. can you confirm ?
> > * suka's patchset would benefit from some optimization in init_upid()
> > and dup_struct_pid()
>
> We have found the reason why Suka's patches showed better performance.
> Some time ago I sent a letter saying that `proc_flush_task()` actually
> never worked with his patches - that's the main problem. After removing
> this call from my patches the results turned to those similar to my.
>
> I'd also like to note that broken-out set of patches is not git bisect
> safe at all. The very first patch of his own OOPSes the node.

FWIW, it's EIP is at `forget_original_parent+0x25` on boot

Process: khelper
exit_notify
do_exit
copy_vm86_regs_to_user
kernel_execve
____call_usermodehelper
...

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

Subject: Re: nptl perf bench and profiling with pidns patchsets
Posted by [Sukadev Bhattiprolu](#) on Sat, 09 Jun 2007 18:47:31 GMT
[View Forum Message](#) <> [Reply to Message](#)

Alexey Dobriyan [adobriyan@sw.ru] wrote:

| On Sat, Jun 09, 2007 at 12:10:25PM +0400, Pavel Emelianov wrote:
| > > * definitely better results for suka's patchset. suka's patchset is
| > > also getting better results with unixbench on a 2.6.22-rc1-mm1 but
| > > the values are really dispersed. can you confirm ?
| > > * suka's patchset would benefit from some optimization in init_upid()
| > > and dup_struct_pid()
| >
| > We have found the reason why Suka's patches showed better performance.
| > Some time ago I sent a letter saying that proc_flush_task() actually
| > never worked with his patches - that's the main problem. After removing
| > this call from my patches the results turned to those similar to my.

i.e with the call removed from both our sets, my patchset is about 1-1.5%
slower than yours ?

| >
| > I'd also like to note that broken-out set of patches is not git bisect
| > safe at all. The very first patch of his own OOPSes the node.

| FWIW, it's EIP is at forget_original_parent+0x25 on boot

| Process: khelper
| exit_notify
| do_exit
| copy_vm86_regs_to_user
| kernel_execve
| ____call_usermodehelper

Thanks for pointing it out. I will backout this change from patch #1 bc
tsk->nsproxy can be null during exit.

```
static inline struct task_struct *child_reaper(struct task_struct *tsk)
{
-   return init_pid_ns.child_reaper;
+   return task_active_pid_ns(tsk)->child_reaper;
}
```

I also fixed the problem in `proc_flush_task()` and am working on fixing signals. After that I will port to more recent kernel and ensure they are bisect safe.

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

Subject: Re: nptl perf bench and profiling with pidns patchsets
Posted by [Cedric Le Goater](#) on Tue, 12 Jun 2007 16:57:55 GMT
[View Forum Message](#) <> [Reply to Message](#)

Pavel Emelianov wrote:

> Cedric Le Goater wrote:

>> Pavel and all,

>

> [snip]

>

>> findings are :

>>

>> * definitely better results for suka's patchset. suka's patchset is

>> also getting better results with unixbench on a 2.6.22-rc1-mm1 but

>> the values are really dispersed. can you confirm ?

>> * suka's patchset would benefit from some optimization in `init_upid()`

>> and `dup_struct_pid()`

>

> We have found the reason why Suka's patches showed better performance.

> Some time ago I sent a letter saying that `proc_flush_task()` actually

> never worked with his patches - that's the main problem. After removing

> this call from my patches the results turned to those similar to my.

>

> I'd also like to note that broken-out set of patches is not git bisect

> safe at all. The very first patch of his own OOPSes the node. Some

> subsequent patches contain misprints that break the compilation, etc.

>

> So I ask you again - let us prepare our patches again and compare the

> performance one more time.

OK. that's fine with me.

I'm not exactly in a neutral zone but I have the blades ready for the next drop of patches. I'll torture them if you don't mind.

C.

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

Subject: Re: nptl perf bench and profiling with pidns patchsets
Posted by [Cedric Le Goater](#) on Wed, 13 Jun 2007 09:25:47 GMT
[View Forum Message](#) <> [Reply to Message](#)

Pavel Emelianov wrote:
> Cedric Le Goater wrote:
>> Pavel Emelianov wrote:
>>> Cedric Le Goater wrote:
>>>> Pavel and all,
>>> [snip]
>>>
>>>> findings are :
>>>>
>>>> * definitely better results for suka's patchset. suka's patchset is
>>>> also getting better results with unixbench on a 2.6.22-rc1-mm1 but
>>>> the values are really dispersed. can you confirm ?
>>>> * suka's patchset would benefit from some optimization in init_upid()
>>>> and dup_struct_pid()
>>> We have found the reason why Suka's patches showed better performance.
>>> Some time ago I sent a letter saying that proc_flush_task() actually
>>> never worked with his patches - that's the main problem. After removing
>>> this call from my patches the results turned to those similar to my.
>>>
>>> I'd also like to note that broken-out set of patches is not git bisect
>>> safe at all. The very first patch of his own OOPSes the node. Some
>>> subsequent patches contain misprints that break the compilation, etc.
>>>
>>> So I ask you again - let us prepare our patches again and compare the
>>> performance one more time.
>> OK. that's fine with me.
>>
>> I'm not exactly in a neutral zone but I have the blades ready for the
>> next drop of patches. I'll torture them if you don't mind.
>
> I do not :) I am going to send my view of pid namespaces this evening
> or tomorrow morning (I am in GMT+3 time zone :)).

I'm in Toulouse, France. GMT+1

> Are you going to fix your patches for comparison?

yes. suka (GMT-8) has a pidns patchset ready for 2.6.22-rc4-mm2 that he should send when he wakes up.

thanks pavel,

C.

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

Subject: Re: nptl perf bench and profiling with pidns patchsets
Posted by [Pavel Emelianov](#) on Wed, 13 Jun 2007 09:27:29 GMT
[View Forum Message](#) <> [Reply to Message](#)

Cedric Le Goater wrote:

> Pavel Emelianov wrote:

>> Cedric Le Goater wrote:

>>> Pavel and all,

>> [snip]

>>

>>> findings are :

>>>

>>> * definitely better results for suka's patchset. suka's patchset is

>>> also getting better results with unixbench on a 2.6.22-rc1-mm1 but

>>> the values are really dispersed. can you confirm ?

>>> * suka's patchset would benefit from some optimization in init_upid()

>>> and dup_struct_pid()

>> We have found the reason why Suka's patches showed better performance.

>> Some time ago I sent a letter saying that proc_flush_task() actually

>> never worked with his patches - that's the main problem. After removing

>> this call from my patches the results turned to those similar to my.

>>

>> I'd also like to note that broken-out set of patches is not git bisect

>> safe at all. The very first patch of his own OOPSes the node. Some

>> subsequent patches contain misprints that break the compilation, etc.

>>

>> So I ask you again - let us prepare our patches again and compare the

>> performance one more time.

>

> OK. that's fine with me.

>

> I'm not exactly in a neutral zone but I have the blades ready for the

> next drop of patches. I'll torture them if you don't mind.

I do not :) I am going to send my view of pid namespaces this evening or tomorrow morning (I am in GMT+3 time zone :)). Are you going to fix your patches for comparison?

> C.
>

Thanks,
Pavel

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