

Re: [Bug #11342] Linux 2.6.27-rc3: kernel BUG at mm/vmalloc.c – bisected

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-08/msg11334.html>

- *From:* Mike Travis <travis@xxxxxxx>
 - *Date:* Tue, 26 Aug 2008 12:01:07 -0700
-

Linus Torvalds wrote:

On Mon, 25 Aug 2008, Linus Torvalds wrote:

checkstack.pl shows these things as the top problems:

```
0xffffffff80266234 smp_call_function_mask [vmlinux]: 2736
0xffffffff80234747 __build_sched_domains [vmlinux]: 2232
0xffffffff8023523f __build_sched_domains [vmlinux]: 2232
```

Anyway, the reason `smp_call_function_mask` and friends have such `_huge_` stack usages for you is that they contain a `'cpumask_t'` on the stack.

In fact, they contain multiple CPU-masks, each 4k-bits – 512 bytes – in size. And they tend to call each other.

Quite frankly, I don't think we were really ready for 4k CPU's. I'm going to commit this patch to make sure others don't do that many CPU's by mistake. It marks MAXCPU's as being 'broken' so you cannot select it, and also limits the number of CPU's that you `_can_` select to "just" 512.

Right now, 4k cpu's is known broken because of the stack usage. I'm not willing to debug more of these kinds of stack smashers, they're really nasty to work with. I wonder how many other random failures these have been involved with?

This patch also makes the `ifdef` mess in `Kconfig` much cleaner and avoids duplicate definitions by just conditionally suppressing the question and giving higher defaults.

We can enable MAXSMP and raise the CPU limits some time in the future. But that future is not going to be before 2.6.27 – the code simply isn't ready for it.

The reason I picked 512 CPU's as the limit is that we `_used_` to limit things to 255. So it's higher than it used to be, but low enough to still

Re: [Bug #11342] Linux 2.6.27-rc3: kernel BUG at mm/vmalloc.c – bisected

feel safe. Considering that a 4k-bit CPU mask (512 bytes) `_almost_` worked, the 512-bit (64 bytes) masks are almost certainly fine.

Still, sane people should limit their `NR_CPUS` to 8 or 16 or something like that. Very very few people really need the pain of big `NR_CPUS`. Not even "just" 512 CPU's.

Travis, Ingo and Thomas cc'd, since they were involved in the original commit (1184dc2ffe2c8fb9afb766d870850f2c3165ef25) that raised the limit.

Linus

Hi Linus,

[Sorry for the long winded response, but I felt that sufficient background is needed to address this issue.... YOMV :-)]

The need to allow distros to set `NR_CPUS=4096` (and `NODES_SHIFT=9`) is critical to our upcoming SGI systems using what we have been calling "UV". This system will be capable of supporting 4096 cpu threads in a single system image (and 16k cpus/2k nodes is right around the corner). While obviously I cannot divulge too many details, it's sufficient to say there are customers who not only require this extended capability, but are extremely excited about it.

But the nature of some of these system environments is that they will not accept a specially built kernel, but only a kernel that has been built and certified (both from the application standpoint as well as the security standpoint) by standard distributions. And you probably know how extensively these distributions test and certify for many known defects and absolutely require that incoming source changes come from the community supported source bases, primarily yours.

Due to the lead time required to accomplish these certifications, the version of the distributions that will be available when this system releases will be based on 2.6.27. (They will allow patches "post-2.6.27-rc.final" as long as those are committed in the source base.) The two distributions that SGI supports for our customers is SLES (SUSE Linux Enterprise Server) and RHEL (Red Hat Enterprise Linux). [They, of course, are free to run any OS of their choosing, but SGI only provides front line support for those two.]

I started last August to begin analyzing how to accomplish the above goals and where exactly are the hot spots in the kernel that would require attention. It quickly became clear that `cpumask_t` and `nodemask_t` are two variables that are very casually used (along with `NR_CPUS`), because the assumption was that 64 "was more than sufficient" for an upper limit and even extending it to 128 or 255 (254 was the maximum IPI broadcast ID until x2apic), only added a few more bytes here and there.

Re: [Bug #11342] Linux 2.6.27-rc3: kernel BUG at mm/vmalloc.c – bisected

I chose not to introduce too many dramatic changes and instead analyzed every instance where `cpumask_t` and `NR_CPUS` was being used (along with the node counterparts.) An initial proposal was to allow the default stack size to be increased, but this was met with a lot of objections because of the extensive work that was done to bring it to its current size.

So in summary, the goals of the changes that I have been making since last October are:

1. Allow a "typical distro" configured kernel with `NR_CPUS=4096` and `NODES_SHIFT=9` to be booted on an `x86_64` system with 2GB of memory. (Some thought was given to use a 512Mb laptop as the base, but because of other memory bloat from using a 64-bit kernel, that was considered not very useful.)

[Note I frequently use an `allyes` and `allmod` config for testing.]

2. To lessen as much as possible, the impact on memory usage for that same kernel on that same system.

3. To lessen as much as possible, the impact on system performance for that same kernel on that same system. [Which mostly depended on #2.]

I booted the first 4096 cpu kernel last February, and since around March or April, Ingo has been (build and boot) testing the `x86` branch using "MAXSMP" to trigger the increased defaults quite extensively (IIRC, it was somewhere between 75% and 90% of all kernels built.) We here at SGI nightly build 4 trees (`linux-2.6`, `linux-next`, `linux-mm`, `linux-x86`) to insure new checkins don't conflict with changes we've made in the past. Unfortunately, our run testing wasn't sufficient to catch this latest error (and I will be quickly fixing that.)

I will also revisit all the past areas to analyze if there have been other abuses of stack and memory space added since the 4k cpu limit was "certified" as usable and releasable. (See below for an initial survey of size increases between a 512cpu/64node configuration and a 4096cpu/512node configuration.)

So perhaps "MAXSMP" is not needed (or perhaps should be more hidden to reduce accidental uses), but allowing the defaults listed above to be in the standard `x86/Kconfig` insures that the distros can at least attempt certification with the maximally configured kernels for their enterprise editions of Linux.

There are many more changes that will be proposed for the 2.6.28 window. Most certainly your concerns, as well as others, about how to change the current "cpumask paradigm" to be more easily manageable for systems with huge cpu counts, will be visited. (And surely be well discussed. :-)

Thanks,
Mike

linux-2.6: v2.6.27-rc4-176-gb8e6c91

=====
Data (-1 500)
... files 2 vars 1421 all 0 lim 500 unch 0

1 - 512-64-allmodconfig
2 - 4096-512-allmodconfig

.1. .2. ..final..

1671168 +3899392 5570560 +233% irq_desc(.data.cacheline_aligned)
591872 +3899392 4491264 +658% irq_cfg(.data.read_mostly)
76800 +537600 614400 +700% early_node_map(.init.data)
66176 +462336 528512 +698% init_mem_cgroup(.bss)
65536 +458752 524288 +700% boot_pageset(.bss)
63648 +419328 482976 +658% kmalloc_caches(.data.cacheline_aligned)
15328 +61376 76704 +400% def_root_domain(.bss)
10240 +43008 53248 +420% change_point_list(.init.data)
8760 +504 9264 +5% init_task(.data)
8192 +57344 65536 +700% kgdb_info(.bss)
6404 +26880 33284 +419% e820_saved(.bss)
6404 +26880 33284 +419% e820(.bss)
6400 +26880 33280 +420% new_bios(.init.data)
5120 +35840 40960 +700% node_devices(.bss)
5120 +21504 26624 +420% change_point(.init.data)
4160 +29120 33280 +700% cpu_bit_bitmap(.rodata)
4096 +28672 32768 +700% __cpu_pda(.init.data)
3776 +25088 28864 +664% hstates(.bss)
3584 +25088 28672 +700% bootmem_node_data(.init.data)
2560 +10752 13312 +420% overlap_list(.init.data)
2048 +14336 16384 +700% x86_cpu_to_node_map_early_map(.init.data)
2048 +14336 16384 +700% was_in_debug_nmi(.bss)
2048 +14336 16384 +700% rio_devs(.init.data)
2048 +14336 16384 +700% passive_cpu_wait(.bss)
2048 +14336 16384 +700% node_memblk_range(.init.data)
2048 +14336 16384 +700% ints(.init.data)
2048 +14336 16384 +700% cpu_in_kgdb(.bss)
1024 +7168 8192 +700% x86_cpu_to_apicid_early_map(.init.data)
1024 +7168 8192 +700% x86_bios_cpu_apicid_early_map(.init.data)
1024 +1024 2048 +100% pxm_to_node_map(.data)
1024 +7168 8192 +700% nodes_add(.bss)
1024 +7168 8192 +700% nodes(.init.data)
512 +3584 4096 +700% zone_movable_pfn(.init.data)
512 +3584 4096 +700% tvec_base_done(.data)
512 +3584 4096 +700% scal_devs(.init.data)
512 +3584 4096 +700% node_data(.data.read_mostly)
512 +3584 4096 +700% memblk_nodeid(.init.data)
0 +2048 2048 . node_to_pxm_map(.data)

0 +2048 2048 . node_order(.bss)
0 +2048 2048 . node_load(.bss)
0 +2048 2048 . fake_node_to_pxm_map(.init.data)
0 +768 768 . rcu_ctrlblk(.data)
0 +768 768 . rcu_bh_ctrlblk(.data)
0 +768 768 . per_cpu__cpu_info(.data.percpu)
0 +768 768 . boot_cpu_data(.data.read_mostly)
0 +760 760 . per_cpu__phys_domains(.data.percpu)
0 +760 760 . per_cpu__node_domains(.data.percpu)
0 +760 760 . per_cpu__cpu_domains(.data.percpu)
0 +760 760 . per_cpu__core_domains(.data.percpu)
0 +760 760 . per_cpu__allnodes_domains(.data.percpu)
0 +720 720 . top_cpuset(.data)
0 +640 640 . per_cpu__flush_state(.data.percpu)
0 +632 632 . pit_clockevent(.data)
0 +632 632 . per_cpu__lapic_events(.data.percpu)
0 +632 632 . lapic_clockevent(.data)
0 +632 632 . hpet_clockevent(.data)
0 +616 616 . net_dma(.data)
0 +579 579 . do_migrate_pages(.text)
0 +568 568 . irq2(.data)
0 +568 568 . irq0(.data)
0 +528 528 . per_cpu__sched_group_phys(.data.percpu)
0 +528 528 . per_cpu__sched_group_cpus(.data.percpu)
0 +528 528 . per_cpu__sched_group_core(.data.percpu)
0 +528 528 . per_cpu__sched_group_allnodes(.data.percpu)
0 +520 520 . out_of_memory(.text)
0 +520 520 . nohz(.data)
0 +512 512 . tick_broadcast_oneshot_mask(.bss)
0 +512 512 . tick_broadcast_mask(.bss)
0 +512 512 . prof_cpu_mask(.data)
0 +512 512 . per_cpu__local_cpu_mask(.data.percpu)
0 +512 512 . per_cpu__cpu_sibling_map(.data.percpu)
0 +512 512 . per_cpu__cpu_core_map(.data.percpu)
0 +512 512 . nohz_cpu_mask(.bss)
0 +512 512 . mce_device_initialized(.bss)
0 +512 512 . mce_cpus(.bss)
0 +512 512 . marked_cpus(.bss)
0 +512 512 . kmem_cach_cpu_free_init_once(.bss)
0 +512 512 . irq_default_affinity(.data)
0 +512 512 . frozen_cpus(.bss)
0 +512 512 . fallback_doms(.bss)
0 +512 512 . cpu_singlethread_map(.data.read_mostly)
0 +512 512 . cpu_sibling_setup_map(.bss)
0 +512 512 . cpu_present_map(.data.read_mostly)
0 +512 512 . cpu_possible_map(.bss)
0 +512 512 . cpu_populated_map(.data.read_mostly)
0 +512 512 . cpu_online_map(.data.read_mostly)
0 +512 512 . cpu_mask_none(.bss)
0 +512 512 . cpu_mask_all(.data.read_mostly)
0 +512 512 . cpu_isolated_map(.bss)

```
0 +512 512 . cpu_initialized(.data)
0 +512 512 . cpu_callout_map(.bss)
0 +512 512 . cpu_callin_map(.bss)
0 +512 512 . cpu_active_map(.bss)
0 +512 512 . cache_dev_map(.bss)
0 +512 512 . cle_mask(.bss)
0 +512 512 . backtrace_mask(.bss)
2647360 +10283499 12930859 +388% Totals
```

```
=====  
Sections (-1 500)  
... files 2 vars 36 all 0 lim 500 unch 0
```

```
1 - 512-64-allmodconfig  
2 - 4096-512-allmodconfig
```

```
.1. .2. ...final..  
66688274 +10345296 77033570 +15% Total  
38237848 +44031 38281879 <1% .debug_info  
8441752 +1215872 9657624 +14% .bss  
2551715 +3136 2554851 <1% .text  
1737600 +4318720 6056320 +248% .data.cacheline_aligned  
1640096 +6784 1646880 <1% .data.percpu  
1175061 +29104 1204165 +2% .rodata  
1073400 +13712 1087112 +1% .debug_abbrev  
901760 +1392 903152 <1% .debug_ranges  
608192 +3906016 4514208 +642% .data.read_mostly  
302704 +13504 316208 +4% .data  
244896 +792112 1037008 +323% .init.data  
123603298 +20689679 144292977 +16% Totals
```

```
=====  
Text/Data ()  
... files 2 vars 6 all 0 lim 0 unch 0
```

```
1 - 512-64-allmodconfig  
2 - 4096-512-allmodconfig
```

```
.1. .2. ...final..  
2551808 +2048 2553856 <1% TextSize  
1679360 +43008 1722368 +2% DataSize  
8441856 +1216512 9658368 +14% BssSize  
2138112 +798720 2936832 +37% InitSize  
1640448 +6144 1646592 <1% PerCPU  
2383872 +8228864 10612736 +345% OtherSize  
18835456 +10295296 29130752 +54% Totals
```

```
=====  
PerCPU ()  
... files 2 vars 22 all 0 lim 0 unch 0
```

```
1 - 512-64-allmodconfig  
2 - 4096-512-allmodconfig
```

```
.1. .2. ..final..
2048 -2048 . -100% vm_event_states
2048 -2048 . -100% softnet_data
2048 -2048 . -100% init_sched_rt_entity
2048 -2048 . -100% core_domains
0 +2048 2048 . sched_group_core
0 +2048 2048 . node_domains
0 +2048 2048 . lru_add_active_pvecs
0 +2048 2048 . init_rt_rq
0 +2048 2048 . cpu_domains
0 +2048 2048 . cpu_core_map
0 +2048 2048 . cpu_buffer
8192 +6144 14336 +75% Totals
```

```
===== Stack (-1 500)
... files 2 vars 126 all 0 lim 500 unch 0
```

```
1 - 512-64-allmodconfig
2 - 4096-512-allmodconfig
```

```
.1. .2. ..final..
0 +2712 2712 . smp_call_function_mask
0 +1576 1576 . setup_IO_APIC_irq
0 +1576 1576 . move_task_off_dead_cpu
0 +1560 1560 . arch_setup_ht_irq
0 +1560 1560 . __assign_irq_vector
0 +1544 1544 . tick_handle_oneshot_broadcast
0 +1544 1544 . msi_compose_msg
0 +1440 1440 . cpuset_write_resmask
0 +1352 1352 . store_scaling_governor
0 +1352 1352 . cpufreq_add_dev
0 +1320 1320 . cpufreq_update_policy
0 +1312 1312 . store_scaling_min_freq
0 +1312 1312 . store_scaling_max_freq
0 +1176 1176 . threshold_create_device
0 +1128 1128 . setup_IO_APIC
0 +1096 1096 . sched_balance_self
0 +1080 1080 . sched_rt_period_timer
0 +1080 1080 . _cpu_down
0 +1064 1064 . set_ioapic_affinity_irq
0 +1048 1048 . store_interrupt_enable
0 +1048 1048 . setup_timer_IRQ0_pin
0 +1048 1048 . setup_ioapic_dest
0 +1048 1048 . set_msi_irq_affinity
0 +1048 1048 . set_ht_irq_affinity
0 +1048 1048 . native_machine_crash_shutdown
0 +1048 1048 . native_flush_tlb_others
0 +1048 1048 . dmar_msi_set_affinity
0 +1040 1040 . store_threshold_limit
0 +1040 1040 . show_error_count
0 +1040 1040 . acpi_map_lsapic
```

0 +1032 1032 . tick_do_periodic_broadcast
0 +1032 1032 . sched_setaffinity
0 +1032 1032 . native_send_call_func_ipi
0 +1032 1032 . local_cpus_show
0 +1032 1032 . local_cpulist_show
0 +1032 1032 . irq_select_affinity
0 +1032 1032 . irq_complete_move
0 +1032 1032 . irq_affinity_proc_write
0 +1032 1032 . flush_tlb_mm
0 +1032 1032 . flush_tlb_current_task
0 +1032 1032 . fixup_irqs
0 +1032 1032 . create_irq
0 +1024 1024 . uv_vector_allocation_domain
0 +1024 1024 . uv_send_IPI_allbutself
0 +1024 1024 . store_error_count
0 +1024 1024 . physflat_send_IPI_allbutself
0 +1024 1024 . pci_bus_show_cpuaffinity
0 +1024 1024 . move_masked_irq
0 +1024 1024 . flush_tlb_page
0 +1024 1024 . flat_send_IPI_allbutself
0 +784 784 . sd_init_ALLNODES
0 +776 776 . sd_init_SIBLING
0 +776 776 . sd_init_NODE
0 +768 768 . sd_init_MC
0 +768 768 . sd_init_CPU
0 +728 728 . update_flag
0 +696 696 . init_intel_cacheinfo
0 +680 680 . __build_sched_domains
0 +648 648 . thread_return
0 +648 648 . schedule
0 +640 640 . cpuset_attach
0 +616 616 . rebalance_domains
0 +600 600 . select_task_rq_fair
0 +600 600 . cache_add_dev
0 +584 584 . shmem_getpage
0 +568 568 . pdflush
0 +552 552 . tick_notify
0 +552 552 . partition_sched_domains
0 +552 552 . free_sched_groups
0 +552 552 . __percpu_alloc_mask
0 +544 544 . taskstats_user_cmd
0 +536 536 . sched_init_smp
0 +536 536 . pci_device_probe
0 +536 536 . cpuset_common_file_read
0 +536 536 . cpupri_find
0 +536 536 . acpi_processor_ffh_cstate_probe
0 +536 536 . __cpu_disable
0 +520 520 . uv_send_IPI_all
0 +520 520 . tick_do_broadcast
0 +520 520 . smp_call_function
0 +520 520 . show_related_cpus

Re: [Bug #11342] Linux 2.6.27-rc3: kernel BUG at mm/vmalloc.c – bisected

```
0 +520 520 . show_affected_cpus
0 +520 520 . prof_cpu_mask_write_proc
0 +520 520 . physflat_send_IPI_mask
0 +520 520 . physflat_send_IPI_all
0 +520 520 . native_smp_send_reschedule
0 +520 520 . native_send_call_func_single_ipi
0 +520 520 . lapic_timer_broadcast
0 +520 520 . irq_set_affinity
0 +520 520 . flat_vector_allocation_domain
0 +520 520 . flat_send_IPI_all
0 +520 520 . find_lowest_rq
0 +520 520 . cpuset_can_attach
0 +520 520 . cpu_callback
0 +520 520 . compat_sys_sched_setaffinity
0 +520 520 . add_del_listener
0 +512 512 . sys_sched_setaffinity
0 +512 512 . sys_sched_getaffinity
0 +512 512 . run_rebalance_domains
0 +512 512 . ioapic_retrigger_irq
0 +512 512 . generic_processor_info
0 +512 512 . force_quiescent_state
0 +512 512 . destroy_irq
0 +512 512 . default_affinity_write
0 +512 512 . cpu_to_phys_group
0 +512 512 . cpu_to_allnodes_group
0 +512 512 . compat_sys_sched_getaffinity
0 +512 512 . check_preempt_curr_rt
0 +512 512 . assign_irq_vector
0 +92248 92248 +0% Totals
```

===== MemInfo ()

... files 0 vars 0 all 0 lim 0 unch 0

(runtime meminfo not collected.)

--

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>