

Re: hash table sizes

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2003-11/4098.html>

From: Martin J. Bligh (mbligh_at_aracnet.com)

Date: 11/26/03

Date: Wed, 26 Nov 2003 08:17:41 -0800
To: William Lee Irwin III <wli@holomorphy.com>

>> *However, I'm curious as to why this crashes X, as I don't see how this
>> code change makes a difference in practice. I didn't think we had any i386
>> NUMA with memory holes between nodes at the moment, though perhaps the x440
>> does.
>> M.
>> PS. No, I haven't tested my rephrasing of your patch either.
>
> mmap() of framebuffer. It takes the box out, not just X. There are
> holes just below 4GB regardless. This has actually been reported by
> rml and some others.
>
> False positives on pfn_valid() result in manipulations of purported page
> structures beyond the bounds of actual allocated pgdat->node_mem_map[]'s,
> potentially either corrupting memory or accessing areas outside memory's
> limits (the case causing oopsen).*

OK. But the hole from 3.75 - 4GB you're referring to doesn't seem to fall under that definition.

- 1) It still has a valid pfn, though the backing memory itself isn't there.
- 2) It's covered by node_start_pfn ... node_start_pfn + node_spanned_pages, which is what your patch tests for. Maybe not if you have = 4GB per node? Will be if you have more than that.

I agree that pfn_valid absolutely has to be correct. The current definition was, I thought, correct unless we have holes **between** nodes. I was under the impression that no box we had uses that setup, but I guess we might do - were you seeing this on x440 or NUMA-Q?

NUMA-Q looks like this:

```
BIOS-e820: 0000000000000000 - 000000000009fc00 (usable)
BIOS-e820: 0000000000100000 - 00000000e0000000 (usable)
BIOS-e820: 00000000fec00000 - 00000000fec09000 (reserved)
BIOS-e820: 00000000ffe80000 - 0000000100000000 (reserved)
BIOS-e820: 0000000100000000 - 0000000400000000 (usable)
```

Linux-Kernel: Re: hash table sizes

which should create mem_map from 0 – 4GB contiguous for node 0, AFAICS (I believe struct pages are created for reserved areas still). Maybe the srat stuff for x440 does something else.

Your patch is correct, I just don't see that it'll fix the X problem.

```
> diff -prauN linux-2.6.0-test10/include/asm-i386/mmzone.h
pfn_valid-2.6.0-test10/include/asm-i386/mmzone.h
> --- linux-2.6.0-test10/include/asm-i386/mmzone.h 2003-11-23 17:31:56.000000000 -0800
> +++ pfn_valid-2.6.0-test10/include/asm-i386/mmzone.h 2003-11-26 01:40:36.000000000 -0800
> @@ -84,14 +84,30 @@ extern struct pglst_data *node_data[];
> + __zone->zone_start_pfn; \
> })
> #define pmd_page(pmd) (pfn_to_page(pmd_val(pmd) >> PAGE_SHIFT))
> +
> +static inline int pfn_to_nid(unsigned long);
> /*
> - * pfn_valid should be made as fast as possible, and the current definition
> - * is valid for machines that are NUMA, but still contiguous, which is what
> - * is currently supported. A more generalised, but slower definition would
> - * be something like this - mbligh:
> - * ( pfn_to_pgdat(pfn) && ((pfn) < node_end_pfn(pfn_to_nid(pfn))) )
> + * pfn_valid must absolutely be correct, regardless of speed concerns.
> */
> -#define pfn_valid(pfn) ((pfn) < num_physpages)
> +static inline int pfn_valid(unsigned long pfn)
> +{
> + u8 nid = pfn_to_nid(pfn);
> + pg_data_t *pgdat;
> +
> + if (nid < MAX_NUMNODES)
> + pgdat = NODE_DATA(nid);
> + else
> + return 0;
> +
> + if (!pgdat)
> + return 0;
> + else if (pfn < pgdat->node_start_pfn)
> + return 0;
> + else if (pfn - pgdat->node_start_pfn >= pgdat->node_spanned_pages)
> + return 0;
> + else
> + return 1;
> +}
>
> /*
> * generic node memory support, the following assumptions apply:
```

Cool, thanks. I'll try runtesting it.

Linux-Kernel: Re: hash table sizes

M.

—

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

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

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