

[PATCH 05/12] i386: During page table initialization always set the leaf page table entries.

[PATCH 05/12] i386: During page table initialization always set the leaf page table entries.

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-04/msg12082.html>

- *From:* ebiederm@xxxxxxxxxxxxx (Eric W. Biederman)
 - *Date:* Mon, 30 Apr 2007 10:03:27 -0600
-

If we don't set the leaf page table entries it is quite possible that will inherit and incorrect page table entry from the initial boot page table setup in head.S. So we need to redo the effort here, so we pick up PSE, PGE and the like.

Hypervisors like Xen require that their page tables be read-only, which is slightly incompatible with our low identity mappings, however I discussed this with Jeremy he has modified the Xen early set_pte function to avoid problems in this area.

Andi I sent this once a part of the discussion on this issue so you may already have this patch in your queue.

Signed-off-by: Eric W. Biederman <ebiederm@xxxxxxxxxxxxx>

arch/i386/mm/init.c | 52 ++++++-----
1 files changed, 20 insertions(+), 32 deletions(-)

```
diff --git a/arch/i386/mm/init.c b/arch/i386/mm/init.c
index b77a43c..dbe16f6 100644
--- a/arch/i386/mm/init.c
+++ b/arch/i386/mm/init.c
@@ -63,18 +63,18 @@ static pmd_t * __init one_md_table_init(pgd_t *pgd)
pmd_t *pmd_table;

#ifdef CONFIG_X86_PAE
- pmd_table = (pmd_t *) alloc_bootmem_low_pages(PAGE_SIZE);
-
- paravirt_alloc_pd(__pa(pmd_table) >> PAGE_SHIFT);
- set_pgd(pgd, __pgd(__pa(pmd_table) | _PAGE_PRESENT));
- pud = pud_offset(pgd, 0);
- if (pmd_table != pmd_offset(pud, 0))
- BUG();
-#else
+ if (!(pgd_val(*pgd) & _PAGE_PRESENT)) {
+ pmd_table = (pmd_t *) alloc_bootmem_low_pages(PAGE_SIZE);
+
+ }
```

[PATCH 05/12] i386: During page table initialization always set the leaf page table entries.

[PATCH 05/12] i386: During page table initialization always set the leaf page table entries.

```
+ paravirt_alloc_pd(__pa(pmd_table) >> PAGE_SHIFT);
+ set_pgd(pgd, __pgd(__pa(pmd_table) | _PAGE_PRESENT));
+ pud = pud_offset(pgd, 0);
+ if (pmd_table != pmd_offset(pud, 0))
+ BUG();
+ }
+ #endif
pud = pud_offset(pgd, 0);
pmd_table = pmd_offset(pud, 0);
- #endif
-
return pmd_table;
}

@@ -84,7 +84,7 @@ static pmd_t * __init one_md_table_init(pgd_t *pgd)
*/
static pte_t * __init one_page_table_init(pmd_t *pmd)
{
- if (pmd_none(*pmd)) {
+ if (!(pmd_val(*pmd) & _PAGE_PRESENT)) {
pte_t *page_table = (pte_t *) alloc_bootmem_low_pages(PAGE_SIZE);

paravirt_alloc_pt(__pa(page_table) >> PAGE_SHIFT);
@@ -109,7 +109,6 @@ static pte_t * __init one_page_table_init(pmd_t *pmd)
static void __init page_table_range_init (unsigned long start, unsigned long end, pgd_t *pgd_base)
{
pgd_t *pgd;
- pud_t *pud;
pmd_t *pmd;
int pgd_idx, pmd_idx;
unsigned long vaddr;
@@ -120,13 +119,10 @@ static void __init page_table_range_init (unsigned long start, unsigned long end
pgd = pgd_base + pgd_idx;

for ( ; (pgd_idx < PTRS_PER_PGD) && (vaddr != end); pgd++, pgd_idx++) {
- if (!(pgd_val(*pgd) & _PAGE_PRESENT))
- one_md_table_init(pgd);
- pud = pud_offset(pgd, vaddr);
- pmd = pmd_offset(pud, vaddr);
+ pmd = one_md_table_init(pgd);
+ pmd = pmd + pmd_index(vaddr);
for ( ; (pmd_idx < PTRS_PER_PMD) && (vaddr != end); pmd++, pmd_idx++) {
- if (pmd_none(*pmd))
- one_page_table_init(pmd);
+ one_page_table_init(pmd);

vaddr += PMD_SIZE;
}
@@ -159,11 +155,7 @@ static void __init kernel_physical_mapping_init(pgd_t *pgd_base)
pfn = 0;
```

[PATCH 05/12] i386: During page table initialization always set the leaf page table entries.

[PATCH 05/12] i386: During page table initialization always set the leaf page table entries.

```
for (; pgd_idx < PTRS_PER_PGD; pgd++, pgd_idx++) {
- if (!(pgd_val(*pgd) & _PAGE_PRESENT))
- pmd = one_md_table_init(pgd);
- else
- pmd = pmd_offset(pud_offset(pgd, PAGE_OFFSET), PAGE_OFFSET);
-
+ pmd = one_md_table_init(pgd);
if (pfn >= max_low_pfn)
continue;
for (pmd_idx = 0; pmd_idx < PTRS_PER_PMD && pfn < max_low_pfn; pmd++, pmd_idx++) {
@@ -172,12 +164,11 @@ static void __init kernel_physical_mapping_init(pgd_t *pgd_base)
/* Map with big pages if possible, otherwise create normal page tables. */
if (cpu_has_pse) {
unsigned int address2 = (pfn + PTRS_PER_PTE - 1) * PAGE_SIZE + PAGE_OFFSET + PAGE_SIZE-1;
- if (!pmd_present(*pmd)) {
- if (is_kernel_text(address) || is_kernel_text(address2))
- set_pmd(pmd, pfn_pmd(pfn, PAGE_KERNEL_LARGE_EXEC));
- else
- set_pmd(pmd, pfn_pmd(pfn, PAGE_KERNEL_LARGE));
- }
+ if (is_kernel_text(address) || is_kernel_text(address2))
+ set_pmd(pmd, pfn_pmd(pfn, PAGE_KERNEL_LARGE_EXEC));
+ else
+ set_pmd(pmd, pfn_pmd(pfn, PAGE_KERNEL_LARGE));
+
pfn += PTRS_PER_PTE;
} else {
pte = one_page_table_init(pmd);
@@ -185,9 +176,6 @@ static void __init kernel_physical_mapping_init(pgd_t *pgd_base)
for (pte_ofs = 0;
pte_ofs < PTRS_PER_PTE && pfn < max_low_pfn;
pte++, pfn++, pte_ofs++, address += PAGE_SIZE) {
- if (pte_present(*pte))
- continue;
-
if (is_kernel_text(address))
set_pte(pte, pfn_pte(pfn, PAGE_KERNEL_EXEC));
else
--
1.5.1.1.181.g2de0
```

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/>