

[PATCH] CPA: Add statistics about state of direct mapping v3

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-03/msg08480.html>

- *From:* Andi Kleen <andi@xxxxxxxxxxxxxxxx>
 - *Date:* Sat, 22 Mar 2008 10:50:41 +0100
-

[I didn't switch to debugfs because I strongly disagreed with that suggestion. But all the other points you made are addressed.]

CPA: Add statistics about state of direct mapping v3

Add information about the mapping state of the direct mapping to /proc/meminfo. I chose /proc/meminfo because that is where all the other memory statistics are too and it is a generally useful metric even outside debugging situations. A lot of split kernel pages means the kernel will run slower.

This way we can see how many large pages are really used for it and how many are split.

Useful for general insight into the kernel.

v2: Add hotplug locking to 64bit to plug a very obscure theoretical race. 32bit doesn't need it because it doesn't support hotadd for lowmem.

Fix some typos

v3: Rename dpages_cnt

Add CONFIG ifdef for count update as requested by tglx

Expand description

Signed-off-by: Andi Kleen <ak@xxxxxxx>

Signed-off-by: Andi Kleen <andi@xxxxxxxxxxxxxxxx>

```

arch/x86/mm/init_32.c | 2 ++
arch/x86/mm/init_64.c | 2 ++
arch/x86/mm/pageattr.c | 24 ++++++
fs/proc/proc_misc.c | 7 ++++++
include/asm-x86/pgtable.h | 3 +++
5 files changed, 38 insertions(+)

```

Index: linux/arch/x86/mm/init_64.c

=====

[PATCH] CPA: Add statistics about state of direct mapping v3

```
--- linux.orig/arch/x86/mm/init_64.c
+++ linux/arch/x86/mm/init_64.c
@@ -316,9 +316,22 @@ __meminit void early_iounmap(void *addr,
__flush_tlb_all();
}

+static void update_page_count(int level, unsigned long pages)
+{
+#ifdef CONFIG_PROC_FS
+ unsigned long flags;
+ /* Protect against CPA */
+ spin_lock_irqsave(&pgd_lock, flags);
+ direct_pages_count[PG_LEVEL_2M] += pages;
+ spin_unlock_irqrestore(&pgd_lock, flags);
+#endif
+}
+
+static unsigned long __meminit
phys_pmd_init(pmd_t *pmd_page, unsigned long address, unsigned long end)
{
+ unsigned long pages = 0;
+
int i = pmd_index(address);

for (; i < PTRS_PER_PMD; i++, address += PMD_SIZE) {
@@ -335,9 +348,11 @@ phys_pmd_init(pmd_t *pmd_page, unsigned
if (pmd_val(*pmd))
continue;

+ pages++;
set_pte((pte_t *)pmd,
pfn_pte(address >> PAGE_SHIFT, PAGE_KERNEL_LARGE));
}
+ update_page_count(PG_LEVEL_2M, pages);
return address;
}

@@ -356,6 +371,7 @@ phys_pmd_update(pud_t *pud, unsigned lon
static unsigned long __meminit
phys_pud_init(pud_t *pud_page, unsigned long addr, unsigned long end)
{
+ unsigned long pages = 0;
unsigned long true_end = end;
int i = pud_index(addr);

@@ -380,6 +396,7 @@ phys_pud_init(pud_t *pud_page, unsigned
}

if (direct_gbpages) {
+ direct_pages_count[PG_LEVEL_1G]++;
set_pte((pte_t *)pud,
```

[PATCH] CPA: Add statistics about state of direct mapping v3

```
pfn_pte(addr >> PAGE_SHIFT, PAGE_KERNEL_LARGE));
true_end = (addr & PUD_MASK) + PUD_SIZE;
@@ -397,6 +414,8 @@ phys_pud_init(pud_t *pud_page, unsigned
}
__flush_tlb_all();

+ update_page_count(PG_LEVEL_1G, pages);
+
return true_end >> PAGE_SHIFT;
}
```

Index: linux/arch/x86/mm/pageattr.c

```
=====
--- linux.orig/arch/x86/mm/pageattr.c
+++ linux/arch/x86/mm/pageattr.c
@@ -18,6 +18,8 @@
#include <asm/pgalloc.h>
#include <asm/proto.h>

+unsigned long direct_pages_count[PG_LEVEL_NUM];
+
+/*
+ * The current flushing context – we pass it instead of 5 arguments:
+ */
@@ -499,6 +501,12 @@ static int split_large_page(pte_t *kpte,
for (i = 0; i < PTRS_PER_PTE; i++, pfn += pfninc)
set_pte(&pbase[i], pfn_pte(pfn, ref_prot));

+ if (address >= (unsigned long)__va(0) &&
+ address < (unsigned long)__va(end_pfn_map << PAGE_SHIFT)) {
+ direct_pages_count[level]--;
+ direct_pages_count[level - 1] += PTRS_PER_PTE;
+ }
+
+/*
+ * Install the new, split up pagetable. Important details here:
+ */
@@ -948,6 +956,22 @@ bool kernel_page_present(struct page *pa

#endif /* CONFIG_DEBUG_PAGEALLOC */

#ifdef CONFIG_PROC_FS
+int arch_report_meminfo(char *page)
+{
+ int n;
+ n = sprintf(page, "DirectMap4k: %8lu\n"
+ "DirectMap2M: %8lu\n",
+ direct_pages_count[PG_LEVEL_4K],
+ direct_pages_count[PG_LEVEL_2M]);
#ifdef CONFIG_X86_64
+ n += sprintf(page + n, "DirectMap1G: %8lu\n",
```

[PATCH] CPA: Add statistics about state of direct mapping v3

```
+ direct_pages_count[PG_LEVEL_1G]);
+ #endif
+ return n;
+ }
+ #endif
+
+ /*
+  * The testcases use internal knowledge of the implementation that shouldn't
+  * be exposed to the rest of the kernel. Include these directly here.
+  Index: linux/include/asm-x86/pgtable.h
```

```
----- linux.orig/include/asm-x86/pgtable.h
+++ linux/include/asm-x86/pgtable.h
@@ -247,8 +247,11 @@ enum {
PG_LEVEL_4K,
PG_LEVEL_2M,
PG_LEVEL_1G,
+ PG_LEVEL_NUM
};
```

```
+extern unsigned long direct_pages_count[PG_LEVEL_NUM];
+
+ /*
+  * Helper function that returns the kernel pagetable entry controlling
+  * the virtual address 'address'. NULL means no pagetable entry present.
+  Index: linux/arch/x86/mm/init_32.c
```

```
----- linux.orig/arch/x86/mm/init_32.c
+++ linux/arch/x86/mm/init_32.c
@@ -198,6 +198,7 @@ static void __init kernel_physical_mappi
is_kernel_text(addr2))
prot = PAGE_KERNEL_LARGE_EXEC;
```

```
+ direct_pages_count[PG_LEVEL_2M]++;
set_pmd(pmd, pfn_pmd(pfn, prot));
```

```
pfn += PTRS_PER_PTE;
@@ -214,6 +215,7 @@ static void __init kernel_physical_mappi
if (is_kernel_text(addr))
prot = PAGE_KERNEL_EXEC;
```

```
+ direct_pages_count[PG_LEVEL_4K]++;
set_pte(pte, pfn_pte(pfn, prot));
}
end_pfn_map = pfn;
Index: linux/fs/proc/proc_misc.c
```

```
----- linux.orig/fs/proc/proc_misc.c
+++ linux/fs/proc/proc_misc.c
@@ -123,6 +123,11 @@ static int uptime_read_proc(char *page,
return proc_calc_metrics(page, start, off, count, eof, len);
```

[PATCH] CPA: Add statistics about state of direct mapping v3

[PATCH] CPA: Add statistics about state of direct mapping v3

```
}  
  
+int __attribute__((weak)) arch_report_meminfo(char *page)  
+{  
+ return 0;  
+}  
+  
static int meminfo_read_proc(char *page, char **start, off_t off,  
int count, int *eof, void *data)  
{  
@@ -219,6 +224,8 @@ static int meminfo_read_proc(char *page,  
  
len += hugetlb_report_meminfo(page + len);  
  
+ len += arch_report_meminfo(page + len);  
+  
return proc_calc_metrics(page, start, off, count, eof, len);  
#undef K  
}  
--
```

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/>