

## Re: [PATCH] reserve RAM below PHYSICAL\_START

---

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-02/msg15213.html>

---

- *From:* "H. Peter Anvin" <[hpa@xxxxxxxxx](mailto:hpa@xxxxxxxxx)>
  - *Date:* Fri, 29 Feb 2008 10:21:08 -0800
- 

Hi Andrea,

Sorry for the long delay in replying.

I'm trying to grok the use cases for this. In particular, it seems like a particularly restricted case of wanting to be able to reserve an arbitrary bit of memory, which seems like it would be more useful (don't we already have `memmap=` options for that, anyway?)

In particular, what's the reason for reserving *\*low\** memory? Low memory (first megabyte) is full of special-use address space which, as the Xen address space discussion has showed, is nontrivial to tamper with even if it initially works. If you want a dedicated chunk of mappable PCI space, it would seem cleaner to have it higher up in the memory map.

Andrea Arcangeli wrote:

Hello,

this patch allows to prevent linux from using the ram below PHYSICAL\_START.

The "reserved RAM" can be mapped by virtualization software with to create a 1:1 mapping between guest physical (bus) address and host physical (bus) address. This will allow pci passthrough with DMA for the guest with current production hardware that misses VT-d. The only detail to take care of is the ram marked "reserved RAM failed". The virtualization software must create for the guest an e820 map that only includes the "reserved RAM" regions but if the guest touches memory with guest physical address in the "reserved RAM failed" ranges (linux guest will do that even if the ram isn't present in the e820 map), it should provide that as ram and map it with a not-ident mapping. This should allow any linux kernel to run fine with pci passthrough and hopefully any other OS too with all VT enabled hardware.

(the virtualization software should do if `(pfn_valid(gfn))`  
`get_page(pfn_to_page(gfn))` instead of `get_user_pages` and equivalent check in the release path)

Re: [PATCH] reserve RAM below PHYSICAL\_START

The trampoline page marked as "reserved RAM failed" can be easily relocated near 640k with an incremental patch to avoid an e820 hole at 0x6000 if any bootloader or OS gets confused.

The end of the patch are just bugfixes. However the limit of the reserved ram is 1G... this can also be relaxed with an incremental patch later on if needed (currently 1G is enough). Perhaps this has other usages.

Let me know if this can be merged, thanks!

```
svm ~ # cat /proc/iomem |head -n 20
00000000-00000fff : reserved RAM failed
00001000-00005fff : reserved RAM
00006000-00007fff : reserved RAM failed
00008000-0009efff : reserved RAM
0009f000-0009ffff : reserved
000cd600-000cffff : pnp 00:0d
000f0000-000fffff : reserved
00100000-0ffffff : reserved RAM
10000000-3dedffff : System RAM
10000000-10329ab2 : Kernel code
10329ab3-104933e7 : Kernel data
104f5000-10558e67 : Kernel bss
3dee0000-3dee2fff : ACPI Non-volatile Storage
3dee3000-3deeffff : ACPI Tables
3def0000-3defffff : reserved
3dff0000-3ffeffff : pnp 00:0d
e0000000-efffffff : reserved
fa000000-fbffffff : PCI Bus #01
fa000000-fbffffff : 0000:01:05.0
fda00000-fdbffffff : PCI Bus #01
svm ~ # hexdump /dev/mem | grep -C2 'cccc cccc cccc cccc'
00007e0 0000 0000 0000 0000 0000 0000 0000 0000 0000
*
0001000 cccc cccc cccc cccc cccc cccc cccc cccc
*
0006000 a5a5 a5a5 8ec8 8ed8 8ec0 66d0 06c7 0000
--
*
0007ff0 0000 0000 0000 0000 3063 1000 0000 0000
0008000 cccc cccc cccc cccc cccc cccc cccc cccc
*
009f000 0002 0000 0000 0000 0000 0000 0000 0000
--
00fffe0 6000 3c03 45e7 0184 0500 0082 01c0 0223
00ffff0 5bea 00e0 31f0 2f32 3931 302f 0037 12fc
0100000 cccc cccc cccc cccc cccc cccc cccc cccc
*
10000000 8d48 f92d ffff 48ff ed81 0000 1000 8948
^C
```

Re: [PATCH] reserve RAM below PHYSICAL\_START

svm ~ #

Signed-off-by: Andrea Arcangeli <andrea@xxxxxxxxxxxx>

```
diff --git a/arch/x86/Kconfig b/arch/x86/Kconfig
--- a/arch/x86/Kconfig
+++ b/arch/x86/Kconfig
@@ -1109,8 +1109,36 @@ config CRASH_DUMP
(CONFIG_RELOCATABLE=y).
For more details see Documentation/kdump/kdump.txt
+config RESERVE_PHYSICAL_START
+bool "Reserve all RAM below PHYSICAL_START (EXPERIMENTAL)"
+depends on !RELOCATABLE && X86_64
+help
+ This makes the kernel use only RAM above __PHYSICAL_START.
+ All memory below __PHYSICAL_START will be left unused and
+ marked as "reserved RAM" in /proc/iomem. The few special
+ pages that can't be relocated at addresses above
+ __PHYSICAL_START and that can't be guaranteed to be unused
+ by the running kernel, will be marked "reserved RAM failed"
+ in /proc/iomem. Those may or may be not used by the kernel
+ (for example smp trampoline pages would only be used if
+ CPU hotplug is enabled).
+
+ The "reserved RAM" can be mapped by virtualization software
+ with /dev/mem to create a 1:1 mapping between guest physical
+ (bus) address and host physical (bus) address. This will
+ allow pci passthrough with DMA for the guest using the ram
+ with the 1:1 mapping. The only detail to take care of is the
+ ram marked "reserved RAM failed". The virtualization
+ software must create for the guest an e820 map that only
+ includes the "reserved RAM" regions but if the guest touches
+ memory with guest physical address in the "reserved RAM
+ failed" ranges (linux guest will do that even if the ram
+ isn't present in the e820 map), it should provide that as
+ ram and map it with a non linear mapping. This should allow
+ any linux kernel to run fine and hopefully any other OS too.
+
config PHYSICAL_START
- hex "Physical address where the kernel is loaded" if (EMBEDDED || CRASH_DUMP)
+ hex "Physical address where the kernel is loaded" if (EMBEDDED || CRASH_DUMP ||
RESERVE_PHYSICAL_START)
default "0x1000000" if X86_NUMAQ
default "0x200000" if X86_64
default "0x100000"
diff --git a/arch/x86/kernel/e820_64.c b/arch/x86/kernel/e820_64.c
--- a/arch/x86/kernel/e820_64.c
+++ b/arch/x86/kernel/e820_64.c
@@ -91,6 +91,11 @@ void __init early_res_to_bootmem(void)
printk(KERN_INFO "early res: %d [%lx-%lx] %s\n", i,
r->start, r->end - 1, r->name);
```

Re: [PATCH] reserve RAM below PHYSICAL\_START

```
reserve_bootmem_generic(r->start, r->end - r->start);
+#ifdef CONFIG_RESERVE_PHYSICAL_START
+ if (r->start < __PHYSICAL_START)
+ add_memory_region(r->start, r->end - r->start,
+ E820_RESERVED_RAM_FAILED);
+#endif
}
}
@@ -231,6 +236,10 @@ void __init e820_reserve_resources(struct
struct resource *data_resource, struct resource *bss_resource)
{
int i;
+#ifdef CONFIG_RESERVE_PHYSICAL_START
+ /* solve E820_RESERVED_RAM vs E820_RESERVED_RAM_FAILED conflicts */
+ update_e820();
+#endif
for (i = 0; i < e820.nr_map; i++) {
struct resource *res;
res = alloc_bootmem_low(sizeof(struct resource));
@@ -238,6 +247,16 @@ void __init e820_reserve_resources(struct
case E820_RAM: res->name = "System RAM"; break;
case E820_ACPI: res->name = "ACPI Tables"; break;
case E820_NVS: res->name = "ACPI Non-volatile Storage"; break;
+#ifdef CONFIG_RESERVE_PHYSICAL_START
+ case E820_RESERVED_RAM_FAILED:
+ res->name = "reserved RAM failed";
+ break;
+ case E820_RESERVED_RAM:
+ memset(__va(e820.map[i].addr),
+ POISON_FREE_INITMEM, e820.map[i].size);
+ res->name = "reserved RAM";
+ break;
+#endif
default: res->name = "reserved";
}
res->start = e820.map[i].addr;
@@ -409,6 +428,12 @@ static void __init e820_print_map(char *
break;
case E820_NVS:
printk(KERN_CONT "(ACPI NVS)\n");
+ break;
+ case E820_RESERVED_RAM:
+ printk(KERN_CONT "(reserved RAM)\n");
+ break;
+ case E820_RESERVED_RAM_FAILED:
+ printk(KERN_CONT "(reserved RAM failed)\n");
break;
default:
printk(KERN_CONT "type %u\n", e820.map[i].type);
@@ -639,9 +664,31 @@ static int __init copy_e820_map(struct e
unsigned long end = start + size;
```

Re: [PATCH] reserve RAM below PHYSICAL\_START

```
unsigned long type = biosmap->type;
+#ifdef CONFIG_RESERVE_PHYSICAL_START
+ /* make space for two more low-prio types */
+ type += 2;
+#endif
+
+ /* Overflow in 64 bits? Ignore the memory map. */
+ if (start > end)
+ return -1;
+
+ #ifdef CONFIG_RESERVE_PHYSICAL_START
+ if (type == E820_RAM) {
+ if (end <= __PHYSICAL_START) {
+ add_memory_region(start, size,
+ E820_RESERVED_RAM);
+ continue;
+ }
+ if (start < __PHYSICAL_START) {
+ add_memory_region(start,
+ __PHYSICAL_START-start,
+ E820_RESERVED_RAM);
+ size -= __PHYSICAL_START-start;
+ start = __PHYSICAL_START;
+ }
+ }
+ #endif
+ add_memory_region(start, size, type);
+ } while (biosmap++, --nr_map);
diff --git a/include/asm-x86/e820.h b/include/asm-x86/e820.h
--- a/include/asm-x86/e820.h
+++ b/include/asm-x86/e820.h
@@ -4,10 +4,19 @@
#define E820MAX 128 /* number of entries in E820MAP */
#define E820NR 0x1e8 /* # entries in E820MAP */
+#ifdef CONFIG_RESERVE_PHYSICAL_START
+#define E820_RESERVED_RAM 1
+#define E820_RESERVED_RAM_FAILED 2
+#define E820_RAM 3
+#define E820_RESERVED 4
+#define E820_ACPI 5
+#define E820_NVS 6
+#else
#define E820_RAM 1
#define E820_RESERVED 2
#define E820_ACPI 3
#define E820_NVS 4
+#endif
+#ifndef __ASSEMBLY__
struct e820entry {
diff --git a/include/asm-x86/page_64.h b/include/asm-x86/page_64.h
--- a/include/asm-x86/page_64.h
```

Re: [PATCH] reserve RAM below PHYSICAL\_START

Re: [PATCH] reserve RAM below PHYSICAL\_START

```
+++ b/include/asm-x86/page_64.h
@@ -29,6 +29,7 @@
#define __PAGE_OFFSET _AC(0xffff810000000000, UL)
#define __PHYSICAL_START CONFIG_PHYSICAL_START
+#define __PHYSICAL_OFFSET (__PHYSICAL_START-0x200000)
#define __KERNEL_ALIGN 0x200000
/*
@@ -47,7 +48,7 @@
#define __PHYSICAL_MASK_SHIFT 46
#define __VIRTUAL_MASK_SHIFT 48
-#define KERNEL_TEXT_SIZE (40*1024*1024)
+#define KERNEL_TEXT_SIZE (40*1024*1024+__PHYSICAL_OFFSET)
#define KERNEL_TEXT_START _AC(0xffffffff80000000, UL)
#ifndef __ASSEMBLY__
diff --git a/include/asm-x86/pgtable_64.h b/include/asm-x86/pgtable_64.h
--- a/include/asm-x86/pgtable_64.h
+++ b/include/asm-x86/pgtable_64.h
@@ -140,7 +140,7 @@ static inline void native_pgd_clear(pgd_
#define VMALLOC_START _AC(0xffffc20000000000, UL)
#define VMALLOC_END _AC(0xffffe1ffffffff, UL)
#define VMEMMAP_START _AC(0xffffe20000000000, UL)
-#define MODULES_VADDR _AC(0xffffffff88000000, UL)
+#define MODULES_VADDR (0xffffffff88000000UL+__PHYSICAL_OFFSET)
#define MODULES_END _AC(0xfffffffffff00000, UL)
#define MODULES_LEN (MODULES_END - MODULES_VADDR)
```

--  
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/>

--  
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/>