

# [3/3] -reserved-ram for PCI passthrough without iommu and without paravirt

---

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-03/msg12086.html>

---

- *From:* Andrea Arcangeli <[andrea@xxxxxxxxxxxxx](mailto:andrea@xxxxxxxxxxxxx)>
  - *Date:* Mon, 31 Mar 2008 19:20:44 +0200
- 

Hello,

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. Please let me know if something like this can be merged in -mm (this is the minimal possible change to achieve the feature). The part at the end is more a fix, but it's only required with this applied (unless you want to have a kexec above ~40M).

Here the complete patchset:

<http://marc.info/?l=kvm-devel&m=120698256716369&w=2>  
<http://marc.info/?l=kvm-devel&m=120698299317253&w=2>  
<http://marc.info/?l=kvm-devel&m=120698328617835&w=2>

Note current mainline is buggy so this patch should also applied with -R for the host kernel to boot but hopefully the regression will be solved sooner than later (no reply yet though).

<http://marc.info/?l=kvm-devel&m=120673375913890&w=2>

```
andrea@svm ~ $ cat /proc/iomem | head
00000000-00000fff : reserved RAM failed
00001000-0008ffff : reserved RAM
00090000-00091fff : reserved RAM failed
00092000-0009efff : reserved RAM
0009f000-0009ffff : reserved
000cd600-000cffff : pnp 00:0d
000f0000-000fffff : reserved
00100000-1fffffff : reserved RAM
20000000-3dedffff : System RAM
```

Signed-off-by: Andrea Arcangeli <[andrea@xxxxxxxxxxxxx](mailto:andrea@xxxxxxxxxxxxx)>

```
diff --git a/arch/x86/Kconfig b/arch/x86/Kconfig
--- a/arch/x86/Kconfig
+++ b/arch/x86/Kconfig
```

### [3/3] –reserved–ram for PCI passthrough without iommu and without paravirt

@@ –1107,8 +1107,36 @@ 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);
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;
@@ -410,6 +429,14 @@ static void __init e820_print_map(char *
case E820_NVS:
printk(KERN_CONT "(ACPI NVS)\n");
break;
#ifdef CONFIG_RESERVE_PHYSICAL_START
+ case E820_RESERVED_RAM:
+ printk(KERN_CONT "(reserved RAM)\n");
+ break;
+ case E820_RESERVED_RAM_FAILED:
+ printk(KERN_CONT "(reserved RAM failed)\n");
+ break;
#endif
default:
printk(KERN_CONT "type %u\n", e820.map[i].type);
break;
@@ -639,9 +666,31 @@ static int __init copy_e820_map(struct e
unsigned long end = start + size;
unsigned long type = biosmap->type;

#ifdef CONFIG_RESERVE_PHYSICAL_START

```

[3/3] –reserved–ram for PCI passthrough without iommu and without paravirt

```
+ /* 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
```

[3/3] –reserved–ram for PCI passthrough without iommu and without paravirt

```
--- a/include/asm-x86/page_64.h
+++ 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

/*
@@ -51,7 +52,7 @@
* Kernel image size is limited to 128 MB (see level2_kernel_pgt in
* arch/x86/kernel/head_64.S), and it is mapped here:
*/
-#define KERNEL_IMAGE_SIZE (128*1024*1024)
+#define KERNEL_IMAGE_SIZE (128*1024*1024+__PHYSICAL_OFFSET)
#define KERNEL_IMAGE_START _AC(0xffffffff80000000, UL)

#ifdef __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)

diff --git a/include/asm-x86/smp_64.h b/include/asm-x86/smp_64.h
--- a/include/asm-x86/smp_64.h
+++ b/include/asm-x86/smp_64.h
@@ -47,7 +47,11 @@ static inline int cpu_present_to_apicid(

#ifdef CONFIG_SMP

+#ifndef CONFIG_RESERVE_PHYSICAL_START
#define SMP_TRAMPOLINE_BASE 0x6000
+#else
+#define SMP_TRAMPOLINE_BASE 0x90000 /* move it next to 640k */
+#endif

extern int __cpu_disable(void);
extern void __cpu_die(unsigned int cpu);
---
```

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/>