Memory Attribute Aliasing on IA-64

Bjorn Helgaas <bjorn.helgaas@hp.com>

May 4, 2006

Memory Attributes

Itanium supports several attributes for virtual memory references. The attribute is part of the virtual translation, i.e., it is contained in the TLB entry. The ones of most interest to the Linux kernel are:

WB	Write-back (cacheable)
UC	Uncacheable
WC	Write-coalescing

System memory typically uses the WB attribute. The UC attribute is used for memory-mapped I/O devices. The WC attribute is uncacheable like UC is, but writes may be delayed and combined to increase performance for things like frame buffers.

The Itanium architecture requires that we avoid accessing the same page with both a cacheable mapping and an uncacheable mapping[1].

The design of the chipset determines which attributes are supported on which regions of the address space. For example, some chipsets support either WB or UC access to main memory, while others support only WB access.

Memory Map

Platform firmware describes the physical memory map and the supported attributes for each region. At boot-time, the kernel uses the EFI GetMemoryMap() interface. ACPI can also describe memory devices and the attributes they support, but Linux/ia64 currently doesn't use this information.

The kernel uses the efi_memmap table returned from GetMemoryMap() to learn the attributes supported by each region of physical address space. Unfortunately, this table does not completely describe the address space because some machines omit some or all of the MMIO regions from the map.

The kernel maintains another table, kern_memmap, which describes the memory Linux is actually using and the attribute for each region. This contains only system memory; it does not contain MMIO space.

The kern_memmap table typically contains only a subset of the system memory described by the efi_memmap. Linux/ia64 can't use all memory in the system because of constraints imposed by the identity mapping scheme.

The efi_memmap table is preserved unmodified because the original boot-time information is required for kexec.

Kernel Identify Mappings

Linux/ia64 identity mappings are done with large pages, currently either 16MB or 64MB, referred to as "granules." Cacheable mappings are speculative[2], so the processor can read any location in the page at any time, independent of the programmer's intentions. This means that to avoid attribute aliasing, Linux can create a cacheable identity mapping only when the entire granule supports cacheable access.

Therefore, kern_memmap contains only full granule-sized regions that can referenced safely by an identity mapping.

Uncacheable mappings are not speculative, so the processor will generate UC accesses only to locations explicitly referenced by software. This allows UC identity mappings to cover granules that are only partially populated, or populated with a combination of UC and WB regions.

User Mappings

User mappings are typically done with 16K or 64K pages. The smaller page size allows more flexibility because only 16K or 64K has to be homogeneous with respect to memory attributes.

Potential Attribute Aliasing Cases

There are several ways the kernel creates new mappings:

mmap of /dev/mem

This uses remap_pfn_range(), which creates user mappings. These mappings may be either WB or UC. If the region being mapped happens to be in kern_memmap, meaning that it may also be mapped by a kernel identity mapping, the user mapping must use the same attribute as the kernel mapping.

If the region is not in kern_memmap, the user mapping should use an attribute reported as being supported in the EFI memory map.

Since the EFI memory map does not describe MMIO on some machines, this should use an uncacheable mapping as a fallback.

mmap of /sys/class/pci_bus/.../legacy_mem

This is very similar to mmap of /dev/mem, except that legacy_mem only allows mmap of the one megabyte "legacy MMIO" area for a specific PCI bus. Typically this is the first megabyte of physical address space, but it may be different on machines with several VGA devices.

"X" uses this to access VGA frame buffers. Using legacy_mem rather than /dev/mem allows multiple instances of X to talk to different VGA cards.

The /dev/mem mmap constraints apply.

mmap of /proc/bus/pci/.../??.?

This is an MMIO mmap of PCI functions, which additionally may or may not be requested as using the WC attribute.

If WC is requested, and the region in kern_memmap is either WC or UC, and the EFI memory map designates the region as WC, then the WC mapping is allowed.

Otherwise, the user mapping must use the same attribute as the kernel mapping.

read/write of /dev/mem

This uses copy_from_user(), which implicitly uses a kernel identity mapping. This is obviously safe for things in kern_memmap.

There may be corner cases of things that are not in kern_memmap, but could be accessed this way. For example, registers in MMIO space are not in kern_memmap, but could be accessed with a UC mapping. This would not cause attribute aliasing. But registers typically can be accessed only with four-byte or eight-byte accesses, and the copy_from_user() path doesn't allow any control over the access size, so this would be dangerous.

ioremap()

This returns a mapping for use inside the kernel.

If the region is in kern_memmap, we should use the attribute specified there.

If the EFI memory map reports that the entire granule supports WB, we should use that (granules that are partially reserved or occupied by firmware do not appear in kern_memmap).

If the granule contains non-WB memory, but we can cover the region safely with kernel page table mappings, we can use ioremap_page_range() as most other architectures do.

Failing all of the above, we have to fall back to a UC mapping.

Past Problem Cases

mmap of various MMIO regions from /dev/mem by "X" on Intel platforms

The EFI memory map may not report these MMIO regions.

These must be allowed so that X will work. This means that when the EFI memory map is incomplete, every /dev/mem mmap must succeed. It may create either WB or UC user mappings, depending on whether the region is in kern_memmap or the EFI memory map.

mmap of 0x0-0x9FFFF /dev/mem by "hwinfo" on HP sx1000 with VGA enabled

The EFI memory map reports the following attributes:

0x00000-0x9FFFF	WB only	
0xA0000-0xBFFFF	UC only	(VGA frame buffer)
0xC0000-0xFFFFF	WB only	

This mmap is done with user pages, not kernel identity mappings, so it is safe to use WB mappings.

The kernel VGA driver may ioremap the VGA frame buffer at 0xA0000, which uses a granule-sized UC mapping. This granule will cover some WB-only memory, but since UC is non-speculative, the processor will never generate an uncacheable reference to the WB-only areas unless the driver explicitly touches them.

mmap of 0x0-0xFFFFF legacy_mem by "X"

If the EFI memory map reports that the entire range supports the same attributes, we can allow the mmap (and we will prefer WB if supported, as is the case with HP sx[12]000 machines with VGA disabled).

If EFI reports the range as partly WB and partly UC (as on sx[12]000 machines with VGA enabled), we must fail the mmap because there's no safe attribute to use.

If EFI reports some of the range but not all (as on Intel firmware that doesn't report the VGA frame buffer at all), we should fail the mmap and force the user to map just the specific region of interest.

mmap of 0xA0000-0xBFFFF legacy_mem by "X" on HP sx1000 with VGA disabled

The EFI memory map reports the following attributes:

```
0x00000-0xFFFFF WB only (no VGA MMIO hole)
```

This is a special case of the previous case, and the mmap should fail for the same reason as above.

read of /sys/devices/.../rom

For VGA devices, this may cause an ioremap() of 0xC0000. This used to be done with a UC mapping, because the VGA frame buffer at 0xA0000 prevents use of a WB granule. The UC mapping causes an MCA on HP sx[12]000 chipsets.

We should use WB page table mappings to avoid covering the VGA frame buffer.

Notes

[1] SDM rev 2.2, vol 2, sec 4.4.1. [2] SDM rev 2.2, vol 2, sec 4.4.6.