DMA attributes

This document describes the semantics of the DMA attributes that are defined in linux/dma-mapping.h.

DMA_ATTR_WEAK_ORDERING

DMA_ATTR_WEAK_ORDERING specifies that reads and writes to the mapping may be weakly ordered, that is that reads and writes may pass each other.

Since it is optional for platforms to implement DMA_ATTR_WEAK_ORDERING, those that do not will simply ignore the attribute and exhibit default behavior.

DMA_ATTR_WRITE_COMBINE

DMA_ATTR_WRITE_COMBINE specifies that writes to the mapping may be buffered to improve performance.

Since it is optional for platforms to implement DMA_ATTR_WRITE_COMBINE, those that do not will simply ignore the attribute and exhibit default behavior.

DMA_ATTR_NO_KERNEL_MAPPING

DMA_ATTR_NO_KERNEL_MAPPING lets the platform to avoid creating a kernel virtual mapping for the allocated buffer. On some architectures creating such mapping is non-trivial task and consumes very limited resources (like kernel virtual address space or dma consistent address space). Buffers allocated with this attribute can be only passed to user space by calling dma_mmap_attrs(). By using this API, you are guaranteeing that you won't dereference the pointer returned by dma_alloc_attr(). You can treat it as a cookie that must be passed to dma_mmap_attrs() and dma_free_attrs(). Make sure that both of these also get this attribute set on each call.

Since it is optional for platforms to implement DMA_ATTR_NO_KERNEL_MAPPING, those that do not will simply ignore the attribute and exhibit default behavior.

DMA_ATTR_SKIP_CPU_SYNC

By default dma_map_{single,page,sg} functions family transfer a given buffer from CPU domain to device domain. Some advanced use cases might require sharing a buffer between more than one device. This requires having a mapping created separately for each device and is usually performed by calling dma_map_{single,page,sg} function more than once for the given buffer with device pointer to each device taking part in the buffer sharing. The first call transfers a buffer from 'CPU' domain to 'device' domain, what synchronizes CPU caches for the given region (usually it means that the cache has been flushed or invalidated depending on the dma direction). However, next calls to dma_map_{single,page,sg}() for other devices will perform exactly the same synchronization operation on the CPU cache. CPU cache synchronization might be a time consuming operation, especially if the buffers are large, so it is highly recommended to avoid it if possible. DMA_ATTR_SKIP_CPU_SYNC allows platform code to skip synchronization of the CPU cache for the given buffer assuming that it has been already transferred to 'device' domain. This attribute can be also used for dma_unmap_{single,page,sg} functions family to force buffer to stay in device domain after releasing a mapping for it. Use this attribute with care!

DMA_ATTR_FORCE_CONTIGUOUS

By default DMA-mapping subsystem is allowed to assemble the buffer allocated by dma_alloc_attrs() function from individual pages if it can be mapped as contiguous chunk into device dma address space. By specifying this attribute the allocated buffer is forced to be contiguous also in physical memory.

DMA_ATTR_ALLOC_SINGLE_PAGES

This is a hint to the DMA-mapping subsystem that it's probably not worth the time to try to allocate memory to in a way that gives better TLB efficiency (AKA it's not worth trying to build the mapping out of larger pages). You might want to specify this if:

- You know that the accesses to this memory won't thrash the TLB. You might know that the accesses are likely to be sequential or that they aren't sequential but it's unlikely you'll ping-pong between many addresses that are likely to be in different physical pages.
- You know that the penalty of TLB misses while accessing the memory will be small enough to be inconsequential. If you are doing a heavy operation like decryption or decompression this might be the case.
- You know that the DMA mapping is fairly transitory. If you expect the mapping to have a short lifetime then it may be worth it to optimize allocation (avoid coming up with large pages) instead of getting the slight performance win of larger pages.

Setting this hint doesn't guarantee that you won't get huge pages, but it means that we won't try quite as hard to get them.

Note

At the moment DMA_ATTR_ALLOC_SINGLE_PAGES is only implemented on ARM, though ARM64 patches will likely be posted soon.

DMA_ATTR_NO_WARN

This tells the DMA-mapping subsystem to suppress allocation failure reports (similarly to __GFP_NOWARN).

On some architectures allocation failures are reported with error messages to the system logs. Although this can help to identify and debug problems, drivers which handle failures (eg, retry later) have no problems with them, and can actually flood the system logs with error messages that aren't any problem at all, depending on the implementation of the retry mechanism.

So, this provides a way for drivers to avoid those error messages on calls where allocation failures are not a problem, and shouldn't bother the logs.

Note

At the moment DMA_ATTR_NO_WARN is only implemented on PowerPC.

DMA_ATTR_PRIVILEGED

Some advanced peripherals such as remote processors and GPUs perform accesses to DMA buffers in both privileged "supervisor" and unprivileged "user" modes. This attribute is used to indicate to the DMA-mapping subsystem that the buffer is fully accessible at the elevated privilege level (and ideally inaccessible or at least read-only at the lesser-privileged levels).