Memory Management

Complete virtual memory map with 4-level page tables

Note

- Negative addresses such as "-23 TB" are absolute addresses in bytes, counted down from the top of the 64-bit address space. It's easier to understand the layout when seen both in absolute addresses and in distance-from-top notation.

  For example 0xffffe90000000000 == -23 TB, it's 23 TB lower than the top of the 64-bit address space (ffffffffffffffff).

  Note that as we get closer to the top of the address space, the notation changes from TB to GB and then MB/KB.

- "16M TB" might look weird at first sight, but it's an easier way to visualize size notation than "16 EB", which few will recognize at first sight as 16 exabytes. It also shows it nicely how incredibly large 64-bit address space is.

Start addr	Offset	End addr	Size	VM area description
0000000000000000	0	00007fffffffffff	128 TB	user-space virtual memory, different per mm
0000800000000000	+128 TB	ffff7fffffffffff	~16M TB	... huge, almost 64 bits wide hole of non-canonical virtual memory addresses up to the -128 TB starting offset of kernel mappings.
				Kernel-space virtual memory, shared between all pr
ffff800000000000	-128 TB	ffff87ffffffffff	8 TB	... guard hole, also reserved for hypervisor
ffff880000000000	-120 TB	ffff887fffffffff	0.5 TB	LDT remap for PTI
ffff888000000000	-119.5 TB	ffffc87fffffffff	64 TB	direct mapping of all physical memory (page_offset
ffffc88000000000	-55.5 TB	ffffc8ffffffffff	0.5 TB	... unused hole
ffffc90000000000	-55 TB	ffffe8ffffffffff	32 TB	vmalloc/ioremap space (vmalloc_base)
ffffe90000000000	-23 TB	ffffe9ffffffffff	1 TB	... unused hole
ffffea0000000000	-22 TB	ffffeaffffffffff	1 TB	virtual memory map (vmemmap_base)
ffffeb0000000000	-21 TB	ffffebffffffffff	1 TB	... unused hole
ffffec0000000000	-20 TB	fffffbffffffffff	16 TB	KASAN shadow memory
				Identical layout to the 56-bit one from here on:
fffffc0000000000	-4 TB	fffffdffffffffff	2 TB	... unused hole vaddr_end for KASLR
fffffe0000000000	-2 TB	fffffe7fffffffff	0.5 TB	cpu_entry_area mapping
fffffe8000000000	-1.5 TB	fffffeffffffffff	0.5 TB	... unused hole
ffffff0000000000	-1 TB	ffffff7fffffffff	0.5 TB	%esp fixup stacks
ffffff8000000000	-512 GB	ffffffeeffffffff	444 GB	... unused hole
ffffffef00000000	-68 GB	fffffffeffffffff	64 GB	EFI region mapping space
fffffffff0000000	-4 GB	ffffffff7fffffff	2 GB	... unused hole
ffffffff80000000	-2 GB	ffffffff9fffffff	512 MB	kernel text mapping, mapped to physical address 0
ffffffff80000000	-2048 MB			
ffffffffa0000000	-1536 MB	fffffffffeffffff	1520 MB	module mapping space
fffffffff0000000	-16 MB			
FIXADDR_START	~-11 MB	fffffffffff5fffff	~0.5 MB	kernel-internal fixmap range, variable size and of
fffffffffe600000	-10 MB	fffffffffe600fff	4 kB	legacy vsyscall ABI
fffffffffe00000	-2 MB	ffffffffffffffff	2 MB	... unused hole

Complete virtual memory map with 5-level page tables

Note

- With 56-bit addresses, user-space memory gets expanded by a factor of 512x, from 0.125 PB to 64 PB. All kernel mappings shift down to the -64 PB starting offset and many of the regions expand to support the much larger physical memory supported.

Start addr	Offset	End addr	Size	VM area description
0000000000000000	0	00ffffffffffffff	64 PB	user-space virtual memory, different per mm
0100000000000000	+64 PB	feffffffffffffff	~16K PB	... huge, still almost 64 bits wide hole of non-ca virtual memory addresses up to the -64 PB starting offset of kernel mappings.

				Kernel-space virtual memory, shared between all pr
ff00000000000000	-64	PB	ff0fffffffffffff	4 PB	... guard hole, also reserved for hypervisor
ff10000000000000	-60	PB	ff10ffffffffffff	0.25 PB	LDT remap for PTI
ff11000000000000	-59.75	PB	ff90ffffffffffff	32 PB	direct mapping of all physical memory (page_offset
ff91000000000000	-27.75	PB	ff9fffffffffffff	3.75 PB	... unused hole
ffa0000000000000	-24	PB	ffd1ffffffffffff	12.5 PB	vmalloc/ioremap space (vmalloc_base)
ffd2000000000000	-11.5	PB	ffd3ffffffffffff	0.5 PB	... unused hole
ffd4000000000000	-11	PB	ffd5ffffffffffff	0.5 PB	virtual memory map (vmemmap_base)
ffd6000000000000	-10.5	PB	ffdeffffffffffff	2.25 PB	... unused hole
ffdf000000000000	-8.25	PB	fffffbffffffffff	~8 PB	KASAN shadow memory
					Identical layout to the 47-bit one from here on:
fffffc0000000000	-4	TB	fffffdffffffffff	2 TB	... unused hole
					vaddr_end for KASLR
fffffe0000000000	-2	TB	fffffe7fffffffff	0.5 TB	cpu_entry_area mapping
fffffe8000000000	-1.5	TB	fffffeffffffffff	0.5 TB	... unused hole
ffffff0000000000	-1	TB	ffffff7fffffffff	0.5 TB	%esp fixup stacks
ffffff8000000000	-512	GB	ffffffeeffffffff	444 GB	... unused hole
ffffffef00000000	-68	GB	fffffffeffffffff	64 GB	EFI region mapping space
fffffffff0000000	-4	GB	fffffffff7ffffff	2 GB	... unused hole
ffffffff80000000	-2	GB	ffffffff9fffffff	512 MB	kernel text mapping, mapped to physical address 0
ffffffff80000000	-2048	MB			
ffffffffa0000000	-1536	MB	ffffffffffefffff	1520 MB	module mapping space
ffffffffff000000	-16	MB			
FIXADDR_START	~-11	MB	ffffffffff5fffff	~0.5 MB	kernel-internal fixmap range, variable size and of
ffffffffff600000	-10	MB	ffffffffff600fff	4 kB	legacy vsyscall ABI
ffffffffffe00000	-2	MB	ffffffffffffffff	2 MB	... unused hole

Architecture defines a 64-bit virtual address. Implementations can support less. Currently supported are 48- and 57-bit virtual addresses. Bits 63 through to the most-significant implemented bit are sign extended. This causes hole between user space and kernel addresses if you interpret them as unsigned.

The direct mapping covers all memory in the system up to the highest memory address (this means in some cases it can also include PCI memory holes).

We map EFI runtime services in the 'efi_pgd' PGD in a 64Gb large virtual memory window (this size is arbitrary, it can be raised later if needed). The mappings are not part of any other kernel PGD and are only available during EFI runtime calls.

Note that if CONFIG_RANDOMIZE_MEMORY is enabled, the direct mapping of all physical memory, vmalloc/ioremap space and virtual memory map are randomized. Their order is preserved but their base will be offset early at boot time.

Be very careful vs. KASLR when changing anything here. The KASLR address range must not overlap with anything except the KASAN shadow area, which is correct as KASAN disables KASLR.

For both 4- and 5-level layouts, the STACKLEAK_POISON value in the last 2MB hole: ffffffffff4111