Memory Objects in Zircon

贾越凯

2019/11/22

用户态可见的结构

- Virtual Memory Object (VMO)
- Virtual Memory Address Region (VMAR)

Virtual Memory Object (VMO)

- 一段连续的内存页面，可以将其映射到多个地址空间中
- 用于在进程之间、内核和用户空间之间共享内存

VMO 相关系统调用

- `zx_vmo_create()` : create a new vmo
- `zx_vmo_read()` : read from a vmo
- `zx_vmo_write()` : write to a vmo
- `zx_vmo_get_size()` : obtain the size of a vmo
- `zx_vmo_set_size()` : adjust the size of a vmo
- `zx_vmo_op_range()` : perform an operation on a range of a vmo
- `zx_vmo_set_cache_policy()` : set the caching policy for pages held by a vmo

Virtual Memory Address Region (VMAR)

- 表示一个进程使用的虚拟地址空间
- 可包含多个不相交的子 VMAR
- 每个进程初始创建一个包含整个地址空间的 root VMAR
- 访问权限：(child <= parent)
- 分配出来的的内存区域默认被随机化

VMAR 相关系统调用:

- `zx_vmar_allocate()` : create a new child VMAR
- `zx_vmar_map()` : map a VMO into a process
- `zx_vmar_unmap()` : unmap a memory region from a process
- `zx_vmar_protect()` : adjust memory access permissions
- `zx_vmar_destroy()` : destroy a VMAR and all of its children

内核中的实现

- `VmObject`
  - `VmObjectPaged`
  - `VmObjectPhysical`
- `VmAddressRegion`
- `VmAspace`

VmObject

- 一段连续的内存页面，用于在进程之间、内核和用户空间之间共享内存
- 可以在创建时不进行分配和映射，而在之后的 `VmAddressRegion::CreateVmMapping()` 进行映射
- 子类:
  - `VmObjectPaged` : the main VM object type, holding a list of pages
  - `VmObjectPhysical` : VMO representing a physical range of memory

VmObjectPaged

- `VmObjectPaged(options, pmm_alloc_flags, size, ...)`
  - 构造函数
- `Create(pmm_alloc_flags, options, size) -> vmo`
  - 创建一个 VMO，不分配
- `CreateContiguous(pmm_alloc_flags, size, alignment) -> vmo`
  - 创建一个 VMO，同时分配一段连续物理内存
- `CreateFromWiredPages(data, size, exclusive) -> vmo`：
  - 创建一个 VMO，使用已有的内存数据
- `CommitRange(offset, len)`
  - 实际分配

VmObjectPaged

- `Resize()`
- `Read(ptr, offset, len)`
- `Write(ptr, offset, len)`
- `GetMappingCachePolicy()`

VmAddressRegion

- 表示一个进程使用的虚拟地址空间
- 可包含多个不相交的子 VMAR，类似于 rCore 中的 `MemorySet`

VmAddressRegion

- `CreateRoot(aspace, vmar_flags) -> out`
- `CreateSubVmar(offset, size, align, vmar_flags, name) -> out`
- `CreateVmMapping(mapping_offset, size, align, vmar_flags, vmo, vmo_offset, arch_mmu_flags, name) -> out`:
- `FindRegion(addr)`
- `Unmap(base, size)`
- `Protect(base, size, new_arch_mmu_flags)`

VmAspace

- 内核或用户可使用的地址空间，类似于 rCore 中的 `PageTable`
- 内部使用平台相关的 `ArchVmAspace` 实现

VmAspace

- 成员字段：
  - `vaddr_t base_`：地址空间基址
  - `size_t size_`：地址空间大小
  - `uint32_t flags_`：`TYPE_USER`、`TYPE_KERNEL`、`TYPE_LOW_KERNEL`、`TYPE_GUEST_PHYS`
  - `fbl::RefPtr<VmAddressRegion> root_vmar_`：root VMAR
  - `ArchVmAspace arch_aspace_`：平台相关的地址空间结构

ArchVmAspace

- `X86ArchVmAspace`
- `ArmArchVmAspace`

ArchVmAspace

- `Map(vaddr, phys[], count, mmu_flags) -> mapped`
  - 将 `vaddr` 开始的 `count` 个页面映射到若干个物理地址 `phys`
- `MapContiguous(vaddr, paddr, count, mmu_flags) -> mapped`
  - 将 `vaddr` 开始的 `count` 个页面映射到 `paddr` 开始的连续物理地址空间
- `Unmap(vddr, count) -> unmapped`：
  - 取消映射
- `Protect(vaddr, count, mmu_flag)`：
  - 修改权限
- `Query(vaddr) -> (paddr, mmu_flags)`
  - 查询虚拟地址映射的物理地址和 `mmu_flag`