Android ION overview

ION is the memory manager of Android, it could be used by graphic and multimedia stacks to allocate buffers.

ION include a buffer sharing mechanism between process and drivers.

ION define opaque handles to manage underline buffers.

ION handles are only map in kernel if that is needed by drivers, it help to save logical address space.

In a same way ION handles aren't mmaped by default in userland but all helpers functions are provided.

ION files

In an Android kernel you can found ION files here:

- include/linux/ion.h
- drivers/gpu/ion/*

ION architecture

ION define different type of heaps:

- ION_HEAP_TYPE_SYSTEM : memory allocated via vmalloc
- ION_HEAP_TYPE_SYSTEM_CONTIG: memory allocated via kmalloc
- ION_HEAP_TYPE_CARVEOUT: memory allocated from a prereserved carveout heap, allocations are physically contiguous
- ION_HEAP_TYPE_CHUNK
- ION_HEAP_TYPE_DMA: memory allocated via DMA API

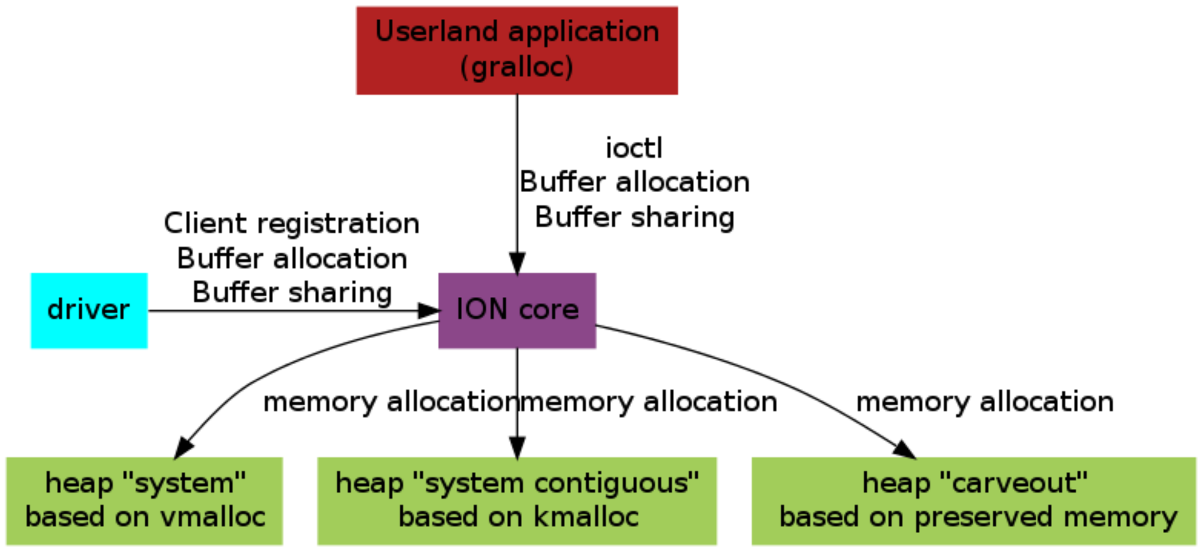

diagram generated from this file

Heaps could be defined in board configuration file with those structures:

```
/**
 * struct ion_platform_heap - defines a heap in the given platform
 * @type:      type of the heap from ion_heap_type enum
 * @id:        unique identifier for heap.  When allocating higher numbers
 *             will be allocated from first.  At allocation these are passed
 *             as a bit mask and therefore can not exceed ION_NUM_HEAP_IDS.
 * @name:      used for debug purposes
 * @base:      base address of heap in physical memory if applicable
 * @size:      size of the heap in bytes if applicable
 * @align:     required alignment in physical memory if applicable
 * @priv:      private info passed from the board file
 *
 * Provided by the board file.
 */
struct ion_platform_heap {
        enum ion_heap_type type;
        unsigned int id;
        const char *name;
        ion_phys_addr_t base;
        size_t size;
        ion_phys_addr_t align;
        void *priv;
};

/**
 * struct ion_platform_data - array of platform heaps passed from board file
 * @nr:        number of structures in the array
 * @heaps:     array of platform_heap structions
 *
 * Provided by the board file in the form of platform data to a platform device.
 */
struct ion_platform_data {
        int nr;
        struct ion_platform_heap *heaps;
};
```

Multiple heaps of each type could be instantiated, and unique ID is used to distinguish and order them.

ION client could select which type of heaps they want to use by setting the correct heap mask when calling ion_alloc. The first heap (i.e. the heap with the lowest ID) is used to do the allocation.

If an ION client want to be more selective it could specify exactly which heap(s) to use (or not) with "flags" parameter of ion_alloc.

ION APIs

Userland API

ioctl are defined to interact with application in userland:

- ION_IOC_ALLOC: allocate memory
- ION_IOC_FREE: free memory
- ION_IOC_MAP: get a file descriptor to mmap
- ION_IOC_SHARE: creates a file descriptor to use to share an allocation
- ION_IOC_IMPORT: imports a shared file descriptor
- ION_IOC_CUSTOM: call architecture specific ion ioctl
- ION_IOC_SYNC - syncs a shared file descriptors to memory

ION_IOC_MAP and ION_IOC_SHARE are using the same code inside ION core.

A suggestion could be to use only ION_IOC_SHARE to avoid any possible confusion with real map or mmap operations.

Kernel API

Kernel drivers can register themself as "client" of ION and specify with a mask which type of heap(s) they want to use:

- ion_client_create: allocate a client and returns it
- ion_client_destroy: free's a client and all it's handles

Buffer allocation and release functions:

- ion_alloc: allocate ion memory, it return an opaque handle it
- ion_free: free a handle

ION use opaque handle to manipulate buffers, drivers must be able to get access to the underline buffer, ION provide lot of function for that:

- ion_phys: returns the physical address and len of a handle
- ion_map_kernel: create mapping for the given handle
- ion_unmap_kernel: destroy a kernel mapping for a handle
- ion_map_dma: create a dma mapping for a given handle, return an sglist
- ion_unmap_dma: destroy a dma mapping for a handle

Instead of share a buffer address between drivers ION allow to use handles, the same mechanism could be used to share buffer between process in useland:

- ion_share: given a handle, obtain a buffer to pass to other clients
- ion_import: given an buffer in another client, import it
- ion_import_fd: given an fd obtained via ION_IOC_SHARE ioctl, import it

Heap API

Heap interface is describe in drivers/gpu/ion/ion_priv.h file.

This API is not exported to drivers or userland.

```
/**
 * struct ion_heap_ops - ops to operate on a given heap
 * @allocate:          allocate memory
 * @free:              free memory
 * @phys               get physical address of a buffer (only define on physically contiguous heaps)
 * @map_dma            map the memory for dma to a scatterlist
 * @unmap_dma          unmap the memory for dma
 * @map_kernel         map memory to the kernel
 * @unmap_kernel       unmap memory to the kernel
 * @map_user           map memory to userspace
 */
```

```
struct ion_heap_ops {
        int (*allocate) (struct ion_heap *heap, struct ion_buffer *buffer, unsigned long len,unsigned long align, unsigned
long flags);
        void (*free) (struct ion_buffer *buffer);
        int (*phys) (struct ion_heap *heap, struct ion_buffer *buffer, ion_phys_addr_t *addr, size_t *len);
        struct scatterlist *(*map_dma) (struct ion_heap *heap, struct ion_buffer *buffer);
        void (*unmap_dma) (struct ion_heap *heap, struct ion_buffer *buffer);
        void * (*map_kernel) (struct ion_heap *heap, struct ion_buffer *buffer);
        void (*unmap_kernel) (struct ion_heap *heap, struct ion_buffer *buffer);
        int (*map_user) (struct ion_heap *mapper, struct ion_buffer *buffer, struct vm_area_struct *vma);
};
```

ION debug

ION provided a debugfs interface in /sys/kernel/debug/ion/ directory.

Each heap has it own debugfs entry where clients memory usage is displayed: /sys/kernel/debug/ion/<<heap name>>

```
$cat /sys/kernel/debug/ion/ion-heap-1
          client              pid             size
        test_ion              2890            16384
```

Each client identify by pid have also a debugfs entry: /sys/kernel/debug/ion/<<pid>> where memory usage displayed.

```
$cat /sys/kernel/debug/ion/2890
       heap_name:    size_in_bytes
      ion-heap-1:    40960 11
```

ION related links

Integrating the ION memory allocator (lwn article by John Stultz)

CategoryCategory

BenjaminGaignard/ion (last modified 2013-09-11 14:56:55)