```
Memory: 1028396K/1044480K available (4482K kernel code, 190K rwdata, 1280K rodata, 204K init, 448K bss, 16084K reserved, 270336K highmem)

Virtual kernel memory layout:

    vector  : 0xffff0000 - 0xffff1000   (   4 kB)

    fixmap  : 0xffc00000 - 0xffe00000   (2048 kB)

    vmalloc : 0xf0000000 - 0xff000000   ( 240 MB)

    lowmem  : 0xc0000000 - 0xef800000   ( 760 MB)

    pkmap   : 0xbfe00000 - 0xc0000000   (   2 MB)

    modules : 0xbf000000 - 0xbfe00000   (  14 MB)

      .text : 0xc0008000 - 0xc05a8d84   (5764 kB)

      .init : 0xc05a9000 - 0xc05dc000   ( 204 kB)

      .data : 0xc05dc000 - 0xc060b980   ( 191 kB)

       .bss : 0xc060b980 - 0xc067bd34   ( 449 kB)
```

0. 对low mamory而言（ ）

paddr = vaddr - PAGE_OFFSET

1. vmalloc 非连续空间的物理映射, VMALLOC_START to VMALLOC_END

2. pkmap空间

```
#define PKMAP_BASE              (PAGE_OFFSET - PMD_SIZE)
```

2M size

在x86体系结构上，高于896MB的所有物理内存的范围大都是高端内存

arch/arm/highmem.c

```
 *
 * Author: Nicolas Pitre
 * Created:     september 8, 2008
 * Copyright:   Marvell Semiconductors Inc.
 *
```

kmap/unkmap系统调用是用来映射高端物理内存页到内核地址空间的api函数，他们分配的内核虚拟地址范围属于[PKMAP_BASE，PAGE_OFFSET]即[0xbfe00000，0xc0000000]范围，大小是2M的虚拟空间，为了映射该块虚拟地址，所使用的二级页表的大小刚好是一个物理page的总计是两个pte table（4KB）

void *kmap(struct page *page)

```
{
        might_sleep();
        if (!PageHighMem(page))
                return page_address(page);
        return kmap_high(page);
}
EXPORT_SYMBOL(kmap);
```

if physical page is highmem, virtual address = kmap_high(page) in pkmap region.

```
/**
 * kmap_high - map a highmem page into memory
 * @page: &struct page to map
 *
 * Returns the page's virtual memory address.
 *
 * We cannot call this from interrupts, as it may block.
 */
void *kmap_high(struct page *page)
{
        unsigned long vaddr;

        /*
         * For highmem pages, we can't trust "virtual" until
         * after we have the lock.
```

```
	 */
	lock_kmap();
	vaddr = (unsigned long)page_address(page);
	if (!vaddr)    为空，表示还未建立virtual-physical mapping
		vaddr = map_new_virtual(page);
	pkmap_count[PKMAP_NR(vaddr)]++;
	BUG_ON(pkmap_count[PKMAP_NR(vaddr)] < 2);
	unlock_kmap();
	return (void*) vaddr;
}


static inline unsigned long map_new_virtual(struct page *page)
{
	unsigned long vaddr;
	int count;
	unsigned int last_pkmap_nr;
	unsigned int color = get_pkmap_color(page);

start:
	count = get_pkmap_entries_count(color);
	/* Find an empty entry */
	for (;;) {
		last_pkmap_nr = get_next_pkmap_nr(color);
		if (no_more_pkmaps(last_pkmap_nr, color)) {
			flush_all_zero_pkmaps();
```

```
                count = get_pkmap_entries_count(color);
        }
        if (!pkmap_count[last_pkmap_nr])
                break;  /* Found a usable entry */
        if (--count)
                continue;

        /*
         * Sleep for somebody else to unmap their entries
         */
        {
                DECLARE_WAITQUEUE(wait, current);
                wait_queue_head_t *pkmap_map_wait =
                        get_pkmap_wait_queue_head(color);

                __set_current_state(TASK_UNINTERRUPTIBLE);
                add_wait_queue(pkmap_map_wait, &wait);
                unlock_kmap();
                schedule();
                remove_wait_queue(pkmap_map_wait, &wait);
                lock_kmap();

                /* Somebody else might have mapped it while we slept */
                if (page_address(page))
                        return (unsigned long)page_address(page);
                /* Re-start */
                goto start;
            }
    }
    vaddr = PKMAP_ADDR(last_pkmap_nr);
    set_pte_at(&init_mm, vaddr,        建立virtual-physical之间的mapping
               &(pkmap_page_table[last_pkmap_nr]), mk_pte(page, kmap_prot));

    pkmap_count[last_pkmap_nr] = 1;
    set_page_address(page, (void *)vaddr);

    return vaddr;
}
```

```
#define PKMAP_ADDR(nr)        (PKMAP_BASE + ((nr) << PAGE_SHIFT))
```

3. module载入空间

modules : 0xbf000000 - 0xbfe00000 (14 MB)

```
root@granite2:~# cat /proc/modules

galcore 161066 0 - Live 0xbf27f000 (O)

ipv6 276100 20 [permanent], Live 0xbf225000

imagepower 1865 0 - Live 0xbf221000 (O)

laservideo_a0 80001 0 - Live 0xbf203000 (O)
```

```
upc 88464 0 - Live 0xbf1c9000 (O)

icetestdriver 15129 0 - Live 0xbf1c0000 (O)

scanblkdriver 52888 0 - Live 0xbf1a8000 (O)

picdriver 55319 0 - Live 0xbf17a000 (O)

dmaalloc 4239 1 laservideo_a0, Live 0xbf175000 (O)

dros 9633 1 laservideo_a0, Live 0xbf16d000 (O)

stepper_api_b0 12711 0 - Live 0xbf165000 (O)

piedriver 179267 0 - Live 0xbf129000 (O)

cisxdriver 34852 0 - Live 0xbf111000 (O)

pegmatite_regulator 4390 4 imagepower,upc,[permanent], Live 0xbf100000

hips_pll 4966 1 laservideo_a0, Live 0xbf0cd000 (O)

m25p80 6875 0 - Live 0xbf0c7000

mv61_cdma 31142 0 - Live 0xbf0ba000

spi_nor 13262 1 m25p80, Live 0xbf0b2000

stepper_mod_b0 16545 1 stepper_api_b0, Live 0xbf076000 (O)

sccplite 5695 0 - Live 0xbf06a000 (O)

ehci_hcd 44353 0 - Live 0xbf048000

spi_pxa2xx_platform 14149 0 - Live 0xbf040000

dcmotor_reg 7724 0 - Live 0xbf028000 (O)

ipc_driver 4578 0 - Live 0xbf023000 (O)

i2c_pxa 7548 0 - Live 0xbf005000

pegmatite_wdt 4332 0 - Live 0xbf000000
```

4. 总体memory map

……

```
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 259600

Kernel command line: console=ttyS0,115200n8 earlyprintk=serial,ttyS0,115200
root=/dev/mmcblk1p2 uio_pdrv_genirq.of_id=generic-uio rootwait

PID hash table entries: 4096 (order: 2, 16384 bytes)

Dentry cache hash table entries: 131072 (order: 7, 524288 bytes)

Inode-cache hash table entries: 65536 (order: 6, 262144 bytes)

Memory: 1028396K/1044480K available (4482K kernel code, 190K rwdata, 1280K rodata, 204K init,
448K bss, 16084K reserved, 270336K highmem)
```

```
 lowmem  : 0xc0000000 - 0xef800000   ( 760 MB)
```

270336K highmem

760 MB + 270M = 1030M

```
pkmap   : 0xbfe00000 - 0xc0000000   (   2 MB)
```

要管理270M high mem？

内核一次最多只能管理2MB的high mem。超过之，则需要临时unmap原来的，mapping新的。