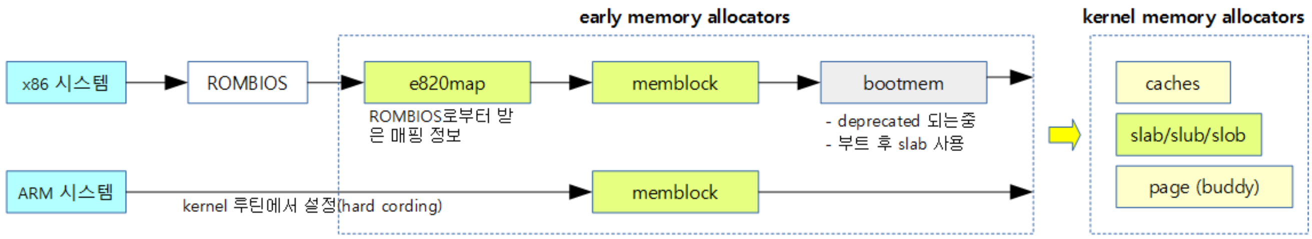
Bootmem with bitmap

2016-02-29 (http://jake.dothome.co.kr/bootmem/) Moon Young-il (http://jake.dothome.co.kr/author/admin/) Linux Kernel (http://jake.dothome.co.kr/category/linux/)

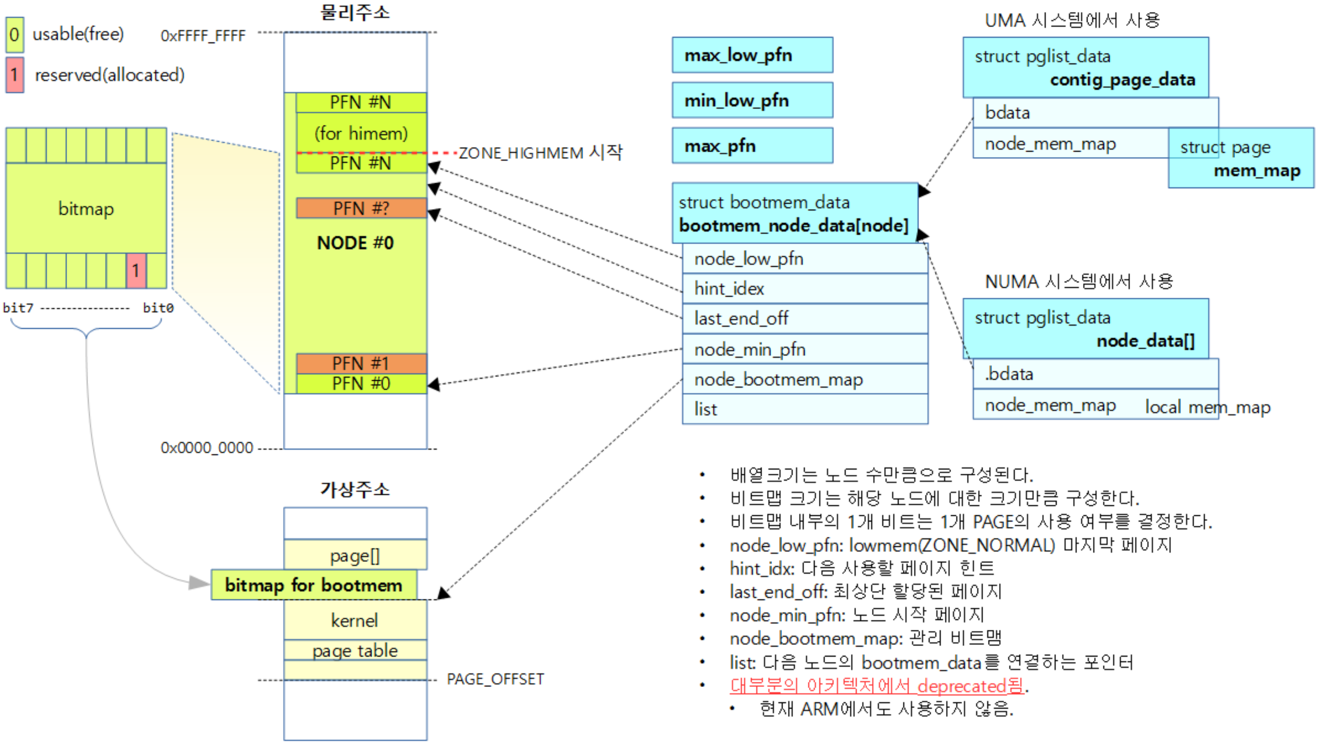
feature

- A boot-time physical memory allocator and configurator
- Bootmem is simple. During the early part of the kernel bootup process, a low-level memory allocator is used by the kernel.
- Bootmem is responsible for allocating the memory required by the system from the initial MMU (paging) function until the buddy allocator is activated during the early boot process, and is converted to buddy allocation after use.
- Bootmem uses a bitmap to indicate whether or not memory is being used.
- With each architecture, Bootmem is increasingly shifting to a method that uses only memblocks to operate instead of being used in the main kernel.
- If you look at the evolution of memory allocation on x86 systems,
  - 1) very very early allocator (early brk model) [x86] – BIOS “e820″을 사용
  - 2) very early allocator (early_res) -> (memblock) [some generic]
    - Kernel 2.6.35 replaces early_res with memblock
  - 3) early allocator (bootmem) [generic]
  - 4) full buddy allocator
- In ARM kernel v3.14-rc1, arm_bootmem_init() has been removed and an CONFIG_NO_BOOTMEM option has been added.



(http://jake.dothome.co.kr/wp-content/uploads/2016/02/bootmem-1.png)

The figure below shows how bootmem is represented as a bitmap.



- 배열크기는 노드 수만큼으로 구성된다.
- 비트맵 크기는 해당 노드에 대한 크기만큼 구성한다.
- 비트맵 내부의 1개 비트는 1개 PAGE의 사용 여부를 결정한다.
- node_low_pfn: lowmem(ZONE_NORMAL) 마지막 페이지
- hint_idx: 다음 사용할 페이지 힌트
- last_end_off: 최상단 할당된 페이지
- node_min_pfn: 노드 시작 페이지
- node_bootmem_map: 관리 비트맵
- list: 다음 노드의 bootmem_data를 연결하는 포인터
- 대부분의 아키텍처에서 deprecated됨.
  - 현재 ARM에서도 사용하지 않음.

(http://jake.dothome.co.kr/wp-content/uploads/2016/02/bootmem-2.png)

Structure

struct bootmem_data

include/linux/bootmem.h

```
#ifndef CONFIG_NO_BOOTMEM
/*
 * node_bootmem_map is a map pointer - the bits represent all physical
 * memory pages (including holes) on the node.
 */
typedef struct bootmem_data {
        unsigned long node_min_pfn;
        unsigned long node_low_pfn;
        void *node_bootmem_map;
        unsigned long last_end_off;
        unsigned long hint_idx;
        struct list_head list;
} bootmem_data_t;

extern bootmem_data_t bootmem_node_data[];
#endif
```

struct pglist_data

include/linux/mmzone.h

```
/*
 * The pg_data_t structure is used in machines with CONFIG_DISCONTIGMEM
 * (mostly NUMA machines?) to denote a higher-level memory zone than the
 * zone denotes.
 *
 * On NUMA machines, each NUMA node would have a pg_data_t to describe
 * it's memory layout.
 *
 * Memory statistics and page replacement data structures are maintained
on a
```

```
 * per-zone basis.
 */
struct bootmem_data;
typedef struct pglist_data {
        struct zone node_zones[MAX_NR_ZONES];
        struct zonelist node_zonelists[MAX_ZONELISTS];
        int nr_zones;
#ifdef CONFIG_FLAT_NODE_MEM_MAP /* means !SPARSEMEM */
        struct page *node_mem_map;
#ifdef CONFIG_PAGE_EXTENSION
        struct page_ext *node_page_ext;
#endif
#endif
#ifndef CONFIG_NO_BOOTMEM
        struct bootmem_data *bdata;
#endif
#ifdef CONFIG_MEMORY_HOTPLUG
        /*
         * Must be held any time you expect node_start_pfn, node_present_pages
         * or node_spanned_pages stay constant.  Holding this will also
         * guarantee that any pfn_valid() stays that way.
         *
         * pgdat_resize_lock() and pgdat_resize_unlock() are provided to
         * manipulate node_size_lock without checking for CONFIG_MEMORY_HOTPLUG.
         *
         * Nests above zone->lock and zone->span_seqlock
         */
        spinlock_t node_size_lock;
#endif
        unsigned long node_start_pfn;
        unsigned long node_present_pages; /* total number of physical pages */
        unsigned long node_spanned_pages; /* total size of physical page
                                             range, including holes */
        int node_id;
        wait_queue_head_t kswapd_wait;
        wait_queue_head_t pfmemalloc_wait;
        struct task_struct *kswapd;     /* Protected by
                                           mem_hotplug_begin/end() */
        int kswapd_max_order;
        enum zone_type classzone_idx;
#ifdef CONFIG_NUMA_BALANCING
        /* Lock serializing the migrate rate limiting window */
        spinlock_t numabalancing_migrate_lock;

        /* Rate limiting time interval */
        unsigned long numabalancing_migrate_next_window;

        /* Number of pages migrated during the rate limiting time interval */
        unsigned long numabalancing_migrate_nr_pages;
#endif
} pg_data_t;
```

contig_page_data etc.

mm/bootmem.c

```
#ifndef CONFIG_NEED_MULTIPLE_NODES
struct pglist_data __refdata contig_page_data = {
        .bdata = &bootmem_node_data[0]
};
EXPORT_SYMBOL(contig_page_data);
#endif
```

```

unsigned long max_low_pfn;
unsigned long min_low_pfn;
unsigned long max_pfn;

bootmem_data_t bootmem_node_data[MAX_NUMNODES] __initdata;

static struct list_head bdata_list __initdata = LIST_HEAD_INIT(bdata_list);
```

consultation

- The NO_BOOTMEM patches (http://lwn.net/Articles/382559/) | LWN.net
- mm: Use memblock interface instead of bootmem (https://lwn.net/Articles/576281/) | LWN.net
- Understanding The Linux Virtual Memory Manager | Mel Gorman – 다운로드 (https://www.kernel.org/doc/gorman/pdf/understand.pdf)
- [Linux] bootmem memory allocator (http://egloos.zum.com/studyfoss/v/5026072) | F/OSS

LEAVE A COMMENT

Your email will not be published. Required fields are marked with *

Comments

name *

email *

Website

WRITE A COMMENT

❮ Inline-Assembly (http://jake.dothome.co.kr/inline-assembly/)

Kmap(Pkmap) ❯ (http://jake.dothome.co.kr/kmap/)