Kernel_4

노트북:	SW		
만든 날짜:	2018-04-10 오후 2:17	수정한 날짜:	2018-04-11 오전 12:39
작성자:	fstopdg		
태그:	:cs find 0		
URL:	http://complicated0idea.tistory.com/14		

2018. 4. 10 화 – 34회차

과정 : TI, DSP, Xilinx Znq FPGA, MCU 기반의 프로그래밍 전문가 과정

Prof. 이상훈
gcccompil3r@gmail.com
Stu. 정상용
fstopdg@gmail.com

Chapter 4 : 메모리관리

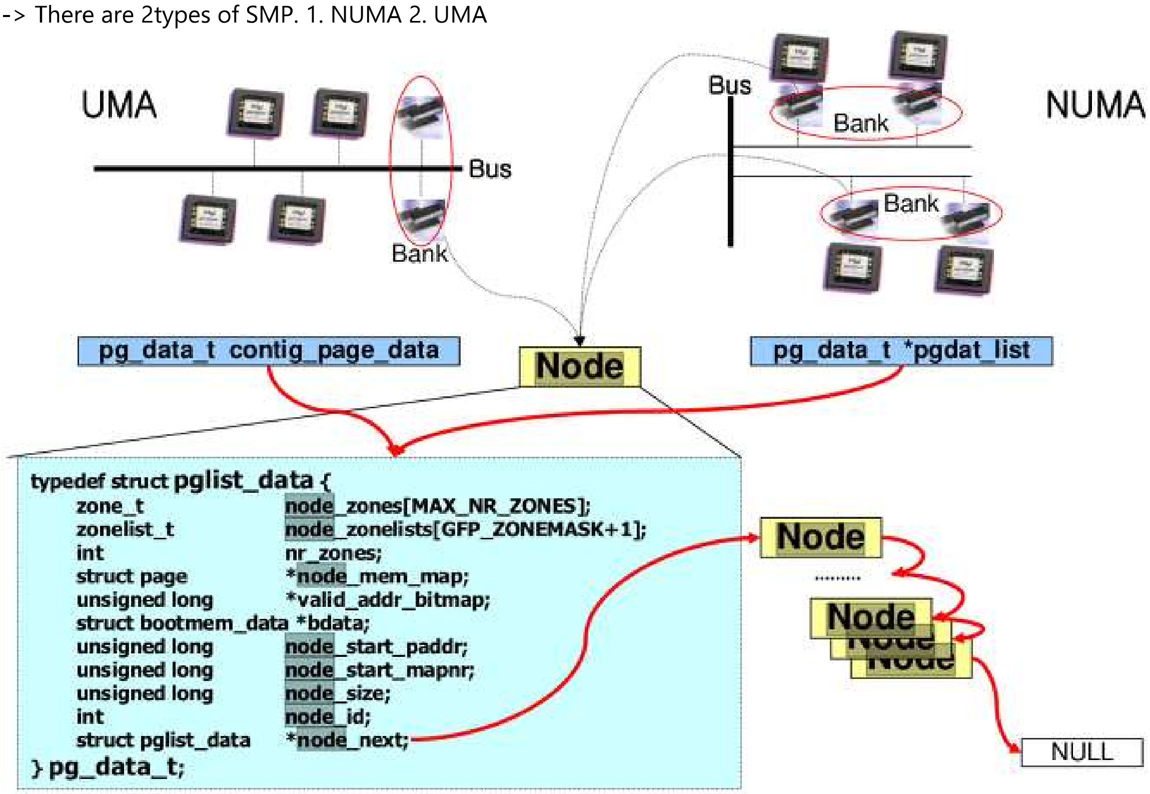
-> The information about memory : mm_struct, slack(when we practiced a lot of programme, we put buf_size like buf[32], buf[64]. slack is about this size), buddy)
-> Virtual Memory, Physical Memory, Paging

1. 메모리 관리 기법과 가상메모리

-> 물리메모리의 한계를 극복하기 위하여 '가상메모리' 개발(i.e Coretex-A series 는 MMU를 달고 나온다. MMU Memory Management Unit)
-> A size of virtual memory : 4GB(32bit)
-> A size of physical memory : 28byte(4byte X 7)
-> The pros : 메모리배치정책이 불필요(kernel이 알아서 해결), 태스크의 빠른 생성이 가능(physical memory를 직접 복사가 아닌 task_struct만 복사하면 되므로), 태스크간 메모리 공유/보호가 쉬움

2. 물리메모리 관리 자료구조(How to work physical memory in linux : Node -> Zone -> Page frame)

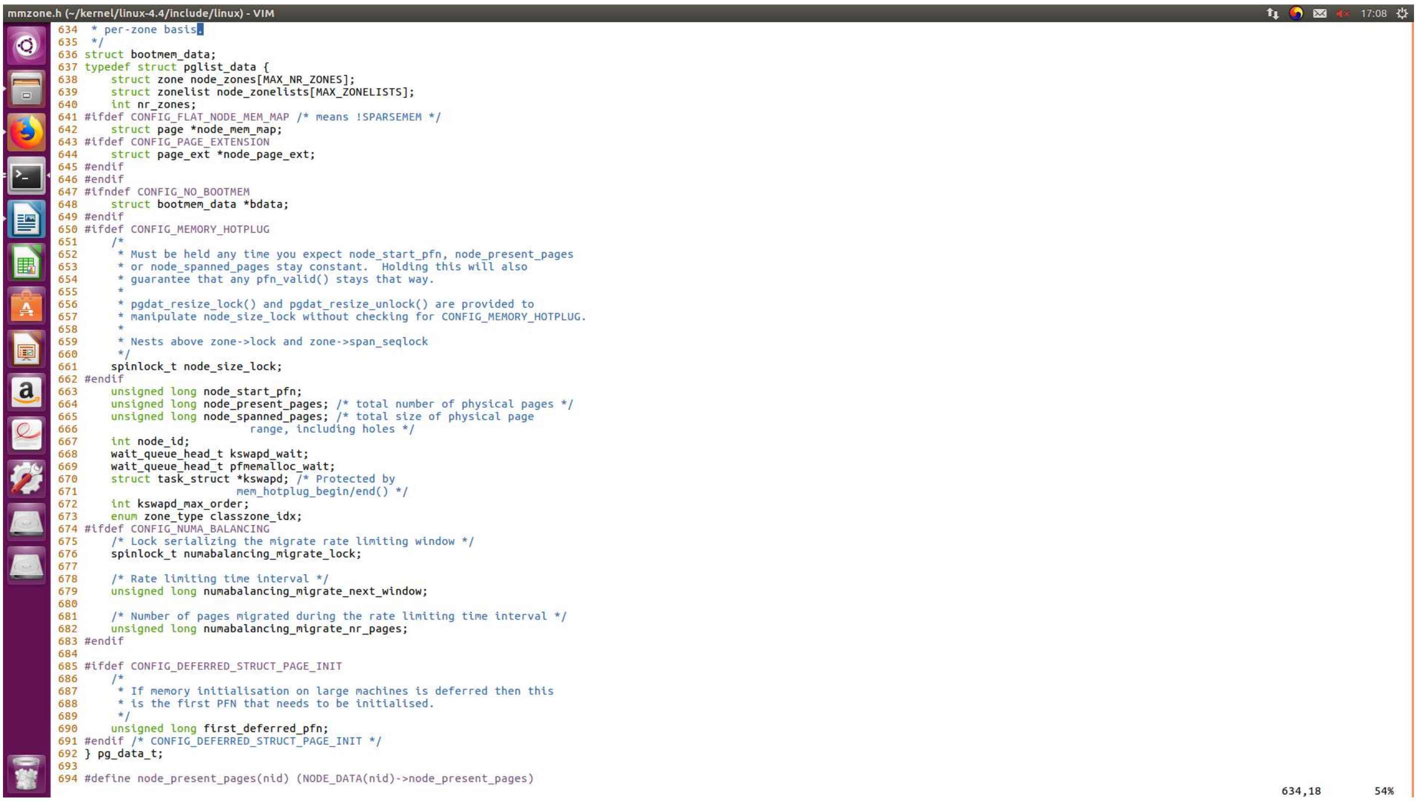
-> Bootloader : 부팅시 물리메모리의 최소공간을 할당해놓음(i.e wait queue를 받을 공간, run queue를 받을 공간등 나중에 사용될 공간을 미리 할당)
-> 2 Purposes : First, How to express a physical memory in linux. Second, Which policies do people use to handle efficiently the limited physical memory.(The policies : paging, buddy, UMA, NUMA...)
-> SMP(Symmetric Multiprocessing) : All CPU and the memories share I/O bus. It causes a bottleneck phenomenon.
-> There are 2types of SMP. 1. NUMA 2. UMA



2.1 Node

-> bank : A group of memories have a same acess velocity.

-> if there is a bank : UMA
-> if there is banks : NUMA
-> **node** is expressed by pglist_data (= pg_data_t) : **bank** struct
i.e A number of node = A number of bank


mmzone.h (~/kernel/linux-4.4/include/linux) - VIM

```c
 * 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

#ifdef CONFIG_DEFERRED_STRUCT_PAGE_INIT
	/*
	 * If memory initialisation on large machines is deferred then this
	 * is the first PFN that needs to be initialised.
	 */
	unsigned long first_deferred_pfn;
#endif /* CONFIG_DEFERRED_STRUCT_PAGE_INIT */
} pg_data_t;

#define node_present_pages(nid)	(NODE_DATA(nid)->node_present_pages)
```


-> a node of UMA : contig_page_data = pglist_data
-> Conclusion : pglist_data cares all node. & pglist_data = bank struct
-> p.98 2nd paragraph : Cache

2.2 Zone
-> It's located in struct pglist_data
-> The information of a zone.
zone : cat /proc/zoneinfo - What we can see : DMA, DM32, Normal, Movable(There is no ISA.)
-> zone : 메모리의 특정한 영역(메모리를 역할에 따라서 나눠주는 역할)

Name	role	Physical memory
ZONE_DMA	DMA stands for Direct Memory Access -> It means DMA is a feature of computer systems that allows certain hardware subsystems to access main system memory independently of the CPU. When a size of data is kind of big, it is used. e.g network, video data, sound data -> V4L2, ALSA	x < 16MB
ZONE_NORMAL	1:1 match	16MB < x <895MB
ZONE_HIGHMEM		896MB < x

-> 16MB이상 : ZONE_NORMAL
-> ZONE_HIGHMEM 896MB기준으로 그 이하 : 1:1로 대용 / 그 이상 : 간접참조(Paging과는 다름 :10bit로 인덱스찾고, 10bit로 인덱스찾고, 12bit로 실제 물리메모리 찾아가는 방식)

2.3 Page frame
-> The smallest unit of memory
-> Page frame is under the struct page
-> This is, each page frame has struct page(therefore, a page's size < a page frame's)

3.Buddy와 Slab

->Q. When we allocate certain data has very small size(30Byte, 60Byte) or too big one compared with the size of page frame(4KB), what happen to page frame?
A. Slab allocator and Buddy allocator solve the promblem(Internal Fragmentation and External Fregmentation).

3.1 버디 할당자(Buddy Allocator : node -> zone -> free_area)
-> it is located on array free_area which is put in struct zone.
it means each free_area has a buddy.

/ include / linux / mmzone.h

```
                            PB_migrate_end, MIGRATETYPE_MASK)

struct free_area {
        struct list_head        free_list[MIGRATE_TYPES];
        unsigned long           nr_free;
};

```

-> Array free_area's index : 0~9(0~9는 해당 free_area가 관리하는 할당의 크기를 나타낸다)
최대할당크기 : 2^10 x 4KB(512 X 4KB
cf
1. How to use ':cs find 0'
when it is impossible use ':cs find 0 dl_rq', type ctag -R on terminal. And, type 'vi -t task_struct' in kernel/linux-4.4
2. http://atsequence.tistory.com/30
3.http://complicated0idea.tistory.com/14