Memory Model -1- (Basic)

2016-03-27 (http://jake.dothome.co.kr/mm-1/) Moon Young-il (http://jake.dothome.co.kr/author/admin/) Linux Kernel (http://jake.dothome.co.kr/category/linux/)

<kernel v5.15>

Physical Memory Model

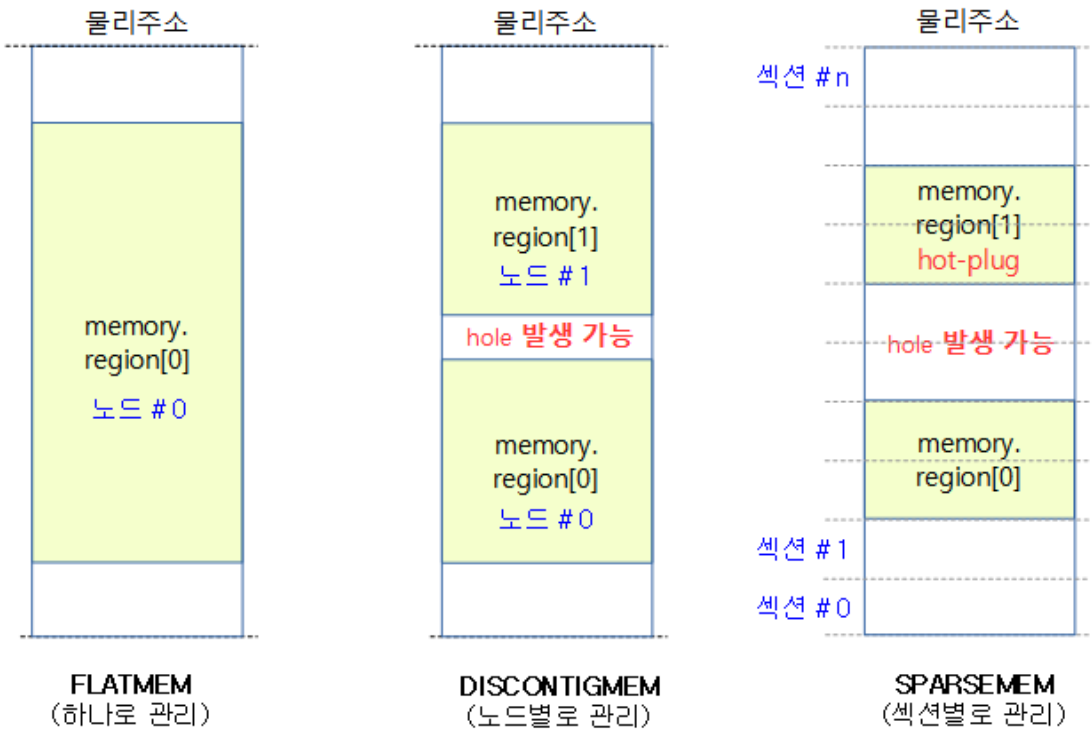
For systems that use consecutive addresses starting with address 0 in physical memory and ending at the end of the address, managing physical memory is the simplest. However, there may be some holes in the middle of such physical memory that are inaccessible to the CPU. In some architectures, when physical memory is placed, a hole may be created in the middle. In addition, NUMA systems can cause large holes between memory banks.

The Linux kernel uses a page structure that manages physical memory in a 4:1 correspondence with a page-frame-by-page (default 1K) basis. For example, in the case of an ARM1 system with 64GB of memory, 1G physical memory is used for 1M (0x100000) page frames, and the corresponding page structure occupies 1M * 64 bytes (default) = 64MB of space. So, about 1.5% of the total physical memory is used by page structures, which are called mem_map.

Physical memory uses the concept of Page Frame Number (PFN) numbers starting from 0 in the physical memory address space, and since the PFN and page structure correspond 1:1, they convert a lot with each other, and the conversion should be easy and fast. The API used to do this is using the pfn_to_page() and page_to_pfn() macro actions. If the entire physical memory is configured without holes, the mem_map composed of page structures is created in batches, and access through PFN can easily point to the corresponding page structure with simple arithmetic operations, but if there are holes in the entire physical memory, the method of mem_map configuration is different, and the Linux kernel provides three physical memory models.

3 Physical Memory Models

LINUX PROVIDES THREE MODELS FOR MANAGING PHYSICAL MEMORY: FLATMEM, DISCONTIGMEM, AND SPARSEMEM, BUT RECENTLY DISCONTIGMEM WAS OMITTED, AND ARCHITECTURES SUPPORT ONE OR MORE OF THESE, AND THE KERNEL BUILDS BY CHOOSING ONE OF THE SUPPORTED MEMORY MODELS. (MOST OF THE 3-BIT SYSTEMS USE FLATMEM, AND THE 32-BIT SYSTEMS USE SPARSEMEM.)



(http://jake.dothome.co.kr/wp-content/uploads/2016/03/mm-1b.png)

Features of each physical memory model

Each physical memory model has a different way of dealing with holes in physical memory. A page structure is required to manage each frame of physical memory, and the aggregate form of these is mem_map. Depending on how they are managed, we will look at the following three physical memory models.

FLATMEM

- The addresses of the installed memory banks are consecutive. In this case, you can choose this model in the simplest way.
- This model can also be used if there is a small hole between the addresses of the installed memory banks. Instead, it wastes as much memory as mem_map corresponding hole.

DISCONTIGMEM

- It was introduced in 2009 for the NUMA system.
- This model was used when there was a large hole between the addresses of the installed memory banks.
- ARM ENDED SUPPORT FOR THIS MODEL IN 2010 WITH THE INTRODUCTION OF SPARSEMEM, A REPLACEMENT FOR DISCONTIGMEM THAT COULD NOT COPE WITH HOT PLUGS.
    - Note: ARM: Remove DISCONTIGMEM support (https://github.com/torvalds/linux/commit/be370302742ff9948f2a42b15cb2ba174d97b930)
- Even in the kernel mainline, this model has been removed entirely.

    - mm: remove CONFIG_DISCONTIGMEM (https://github.com/torvalds/linux/commit/bb1c50d3967f69f413b333713c2718d48d1ab7ea#diff-6a0166b1d8bbb576287048e4de42eb7e8a1f118e357f407d3bdf7da9fc26d94d) (2021, v5.14-rc1)

SPARSEMEM

- This model can be selected if there are large holes between the addresses of installed memory banks (including NUMA systems) or if hotplug memory is required.
    - Note: x86_64: Make sparsemem/vmemmap the default memory model (http://lwn.net/Articles/258305/)
    - x86 and arm64 use the sparsemem memory model and vmemmap kernel options by default.
- Online/offline (hotplug memory) is managed by section, and the section size varies by kernel option or architecture, ranging from tens of MB ~ several G.
    - In the case of arm32, it is different for each kernel option that is configured.
    - In the case of arm64, 1G was used as the default value, but it was recently changed to 128M.
        - Note: arm64/sparsemem: reduce SECTION_SIZE_BITS (https://github.com/torvalds/linux/commit/f0b13ee23241846f6f6cd0d119a8ac8059416ec4#diff-9590f22a80ad3a29d9c97bab548e481c48ac04e5ca5441c9360c69458829634c) (2021, v5.12-rc1)

ARM with NUMA

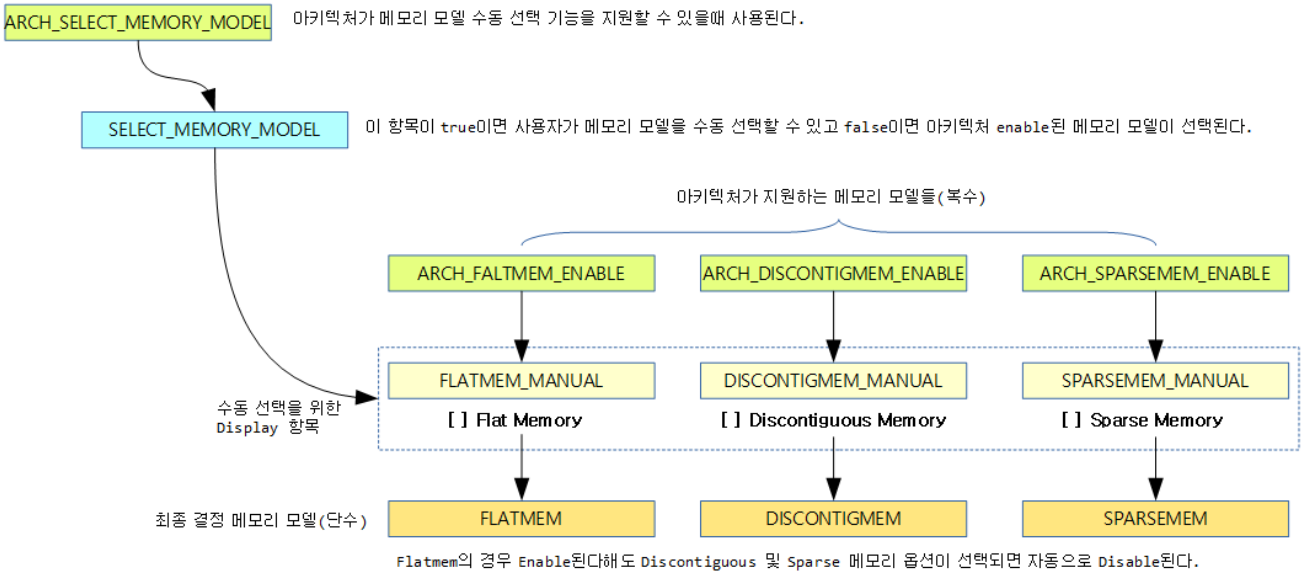
- ARM is not supported in the mainstream kernel.
    - The 2012 RFC patch included support for NUMA in arm, and the linaro kernel also supports NUMA in arm, but it is not included in the mainstream because it is not absolutely necessary.
        - Note: Add support for CONFIG NUMA to ARM (http://elinux.org/Add_support_for_CONFIG_NUMA_to_ARM) | elinux.org
- arm64 was supported by the 2014 NUMA patch, and in 2016 the mainstream kernel v4.7-rc1 was applied by default.
    - See: arm64:numa: Add numa support for arm64 platforms (https://lwn.net/Articles/628280/) | LWN.net
    - Note: arm64, numa: Add NUMA support for arm64 platforms. (https://github.com/torvalds/linux/commit/1a2db300348b799479d2d22b84d51b27ad0458c7#diff-74b704a6995edb0529fc619230b82b94)

아키텍처 vs 메모리모델	FLATMEM	DISCONTIGMEM	SPARSEMEM
arm	O(default)	X	△[1]
arm64	X	X	O(default)
x86_32	O(default)	X	O
x86_32 with NUMA	X	O(default)	O
x86_64	X	X	O(default)
x86_64 with NUMA	X	X	O(default)

[1] SA-1100, RiscPC, RealView-PBX 및 LPAE를 사용하는 일부 머신에서 SPARSEMEM을 사용한다.

(http://jake.dothome.co.kr/wp-content/uploads/2016/03/mm-2a.png)

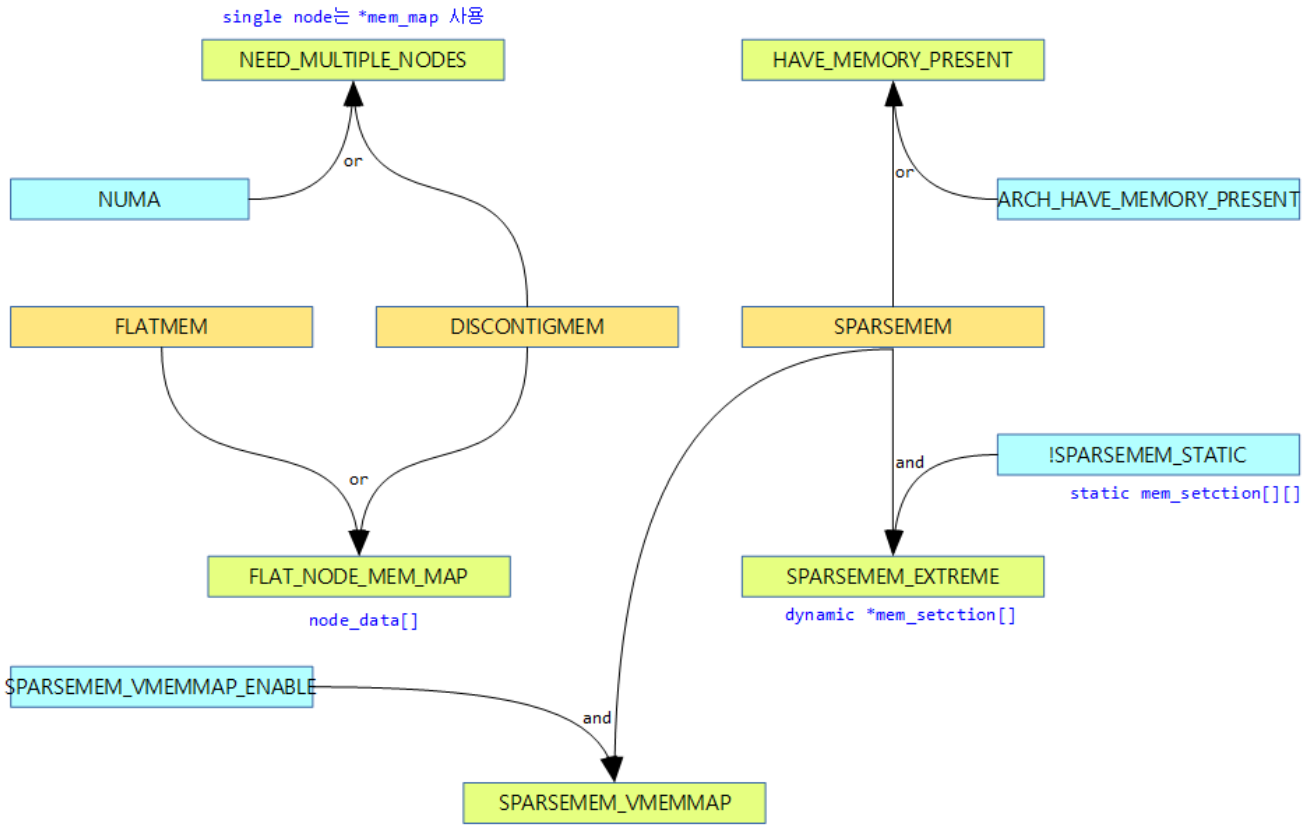
Memory Model Main Kernel Options



(http://jake.dothome.co.kr/wp-content/uploads/2016/03/mm-4.png)

- In the case of ARM64, the following options are used, using only one SPARSEMEM memory model:
  - CONFIG_ARCH_SELECT_MEMORY_MODEL=y
  - CONFIG_SELECT_MEMORY_MODEL=y
  - CONFIG_ARCH_SPARSEMEM_ENABLE=y
  - CONFIG_ARCH_SPARSEMEM_DEFAULT=y

Memory Model Additional Kernel Options



(http://jake.dothome.co.kr/wp-content/uploads/2016/03/mm-5.png)

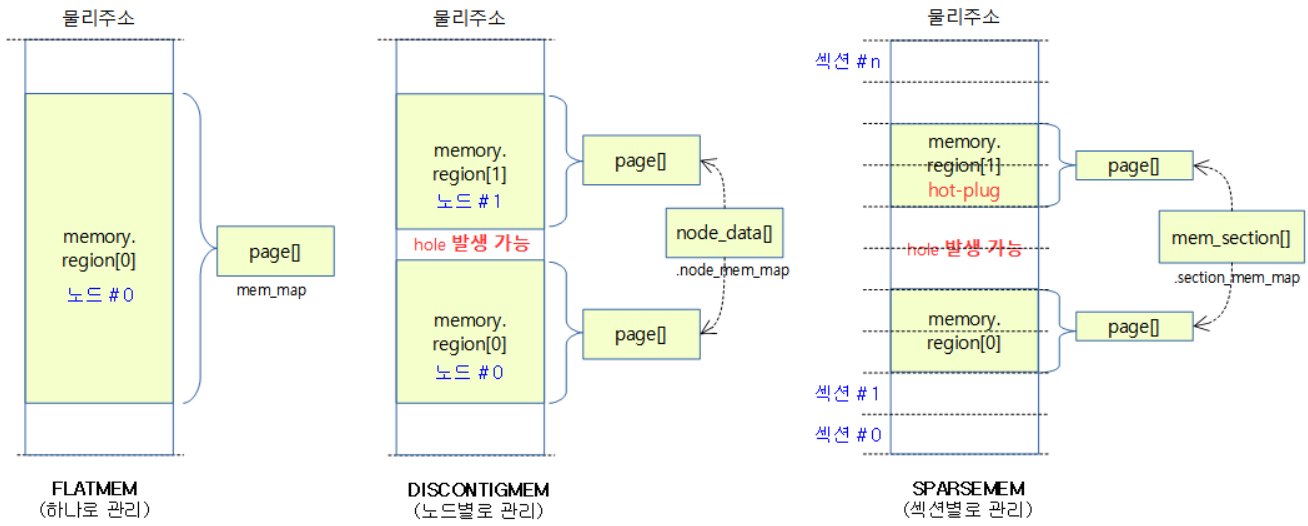
- NEED_MULTIPLE_NODES
  - It is available in NUMA systems ~~or DISCONTIGMEM memory models~~.
  - NODE_DATA() macro to point to the mem_map of each node.
  - Use a global node_data[] array to access the mem_map on a node-by-node basis.
- FLAT_NODE_MEM_MAP
  - It is available in FLATMEM ~~or DISCONTIGMEM~~ memory models.
  - The mem_map pglist_data that describes the node information is accessed through the node_mem_map of the member variables of the structure.
- HAVE_MEMORY_PRESENT
  - For the SPARSEMEM memory model, holes can occur in the requested range, so that the presence of memory can be managed on a per-section basis.
- SPARSEMEM_EXTREME
  - This is one of two choices used when using the sparse memory model.
  - mem_section[] array dynamically allocated and used
  - It is mainly used on 64-bit systems with a large number of sections that need to be created.
- SPARSEMEM_STATIC
  - This is one of two choices used when using the sparse memory model.
  - Use the mem_section[] that you have statically assigned at compile time.
  - It is mainly used on 32-bit systems with a small number of sections to create.
- SPARSEMEM_VMEMMAP
  - You can use vmemmap to quickly convert pfn and page descriptor addresses.
  - vmalloc is used on 64-bit systems with large address ranges.
- MEMORY_HOTPLUG
  - Memory can be added during system operations.

- MEMORY_HOTREMOVE
  - Memory can be removed during system operation.
  - You can choose from MEMORY_HOTPLUG, ARCH_ENABLE_MEMORY_HOTREMOVE, and MIGRATION options when they are available.
- HIGHMEM
  - This is a range of memory that is not pre-mapped 32:1 to the kernel at 1bit, so the kernel must map and use it whenever it needs direct access.
  - On a 32-bit system, if the system memory is larger than the kernel space, the HIGHMEM option can be used to make more memory available.
- HIGHPTE
  - On a 32-bit system, the secondary PTE is assigned to highmem.
  - If a 32-bit system uses more than several gigabytes of memory, it is recommended to load the secondary PTE into highmem so that it does not consume lowmem memory.
- NODE_NOT_IN_PAGE_FLAGS
  - If the node number fails to be recorded in the flags member variable of the page structure, the node number is mapped 1:1 to a separate section and stored
  - Used on 32-bit systems when node numbers cannot be recorded due to insufficient bit digits

Memory Models and Page Management Maps (mem_map)

The page structure array is called mem_map as the page frame of the entire physical memory, and the mem_map is managed for each memory model as follows.

- FLATMEM
  - *mem_map pointer points to an array of page[] structures.
  - Since there is only one node, it uses a global struct contig_page_data, whose member variables node_mem_map also point to an array of page[] structures.
- DISCONTIGMEM
  - Since there is more than one node, we use an array of node_data[], whose member variable node_mem_map points to the array of page[] structures associated with each node.
- SPARSEMEM
  - There are two ways to do this.
    - SPARSEMEM_STATIC
      - The mem_section[] array for the number of sections is placed in static memory at compile time, and the section_mem_map of member variables for each section entry points to an array of page[] structures that manage pages equal to the size of that section.
    - SPARSEMEM_EXTREAM
      - The number of sections is an array of *mem_section[] pointers in static memory, and in fact, each mem_section[] array dynamically allocates a mem_section[] array to use memory when the kernel initializes, and the use of mem_section[] is the same as SPARSEMEM_STATIC.



(http://jake.dothome.co.kr/wp-content/uploads/2016/03/mm-3b.png)

consultation

- Memory Model -1- (Basic) (http://jake.dothome.co.kr/mm-1) | Sentence C – Current post
- Memory Model -2- (mem_map) (http://jake.dothome.co.kr/mem_map) | Qc
- Memory Model -3- (Sparse Memory) (http://jake.dothome.co.kr/sparsemem/) | Qc
- Memory Model -4- (APIs) (http://jake.dothome.co.kr/mem_map_page) | Qc
- ZONE Type (http://jake.dothome.co.kr/zone-types) | Qc
- bootmem_init (http://jake.dothome.co.kr/bootmem_init-64) | Qc
- zone_sizes_init() (http://jake.dothome.co.kr/free_area_init_node/) | Sentence C – Current post
- NUMA -1- (ARM64 initialization) (http://jake.dothome.co.kr/numa-1) | Qc
- build_all_zonelists() (http://jake.dothome.co.kr/build_all_zonelists) | Qc

- Memory: the flat, the discontiguous, and the sparse(2019) (https://lwn.net/Articles/789304/) | LWN.net
- sparsemem memory model (https://lwn.net/Articles/134804/)(2005) | LWN.net
- Sparsemem(2010) (http://mytechkorner.blogspot.kr/2010/12/sparsemem.html)

3 thoughts to “Memory Model -1- (Basic)”

SEUNGHAK LEE
2017-10-11 19:01 (http://jake.dothome.co.kr/mm-1/#comment-173049)

Hello. Thank you so much for posting such great information.
I haven't finished reading it yet, but I'm leaving a question because I have a question.

First of all, I would like to know if you are referring to a memory bank in this article that consists of multiple memory arrays within a
DRAM device, or

simply a bank that represents a set of memory address areas (e.g. cache bank).

Also, if you set it to SPARSEMEM, it says that you can use the Memory Hotplug on a per-section basis, so I wonder what part of the hardware the section refers to.

Speaking of which, I would like to know what unit the Section leads to in the internal structure of the DRAM Device.

RESPONSE (/MM-1/?REPLYTOCOM=173049#RESPOND)

MOON YOUNG-IL (HTTP://JAKE.DOTHOME.CO.KR)
2017-10-11 21:09 (http://jake.dothome.co.kr/mm-1/#comment-173050)

Hello?
Memory banks are logically stated.
Let's take an example of a NUMA system consisting of two nodes:
It is assumed that the DRAM controller used for each node can use up to 2G of maximum physical memory. It also assumes that the physical memory start address starts with 64x0.
0st node: DRAM Controller – manage 1x0 0000_0000 ~ 0000x0 0010 0000 0000nd node: DRAM Controller – manage
2x0 0010 0000 ~ 0000x0 0020 0000

Note that in your CPU architecture or SoC design, you can place the second node next to the 64G space used by the first node, or you can place it at a reasonable distance.

The user plugged in four 4 GB memory on the first node and four 4 GB memory on the second node. In this case, you've configured a total of 4 GB.

If you set up the system like this, you have two logical memory banks that I have represented.
(It should not be thought of in conjunction with the memory bank, memory slot, etc. used by the hardware)

Let me give you another example.
Even if you use one controller, if the architecture design a memory placement exceeds a certain capacity, a certain space will be skipped.
The DDR memory controller manages up to 64G, but when placed in physical memory, it is used as follows, but in the case of arm64, it is spaced out as follows:

0x00 0000 0000 – 0x00 FFFF FFFF ...
0x00 8000 0000 – 0x00 FFFF FFFF (1st bank: 2st bank: placing 0G in the first physical space)
01x0000 0000 0 – 08x7 0FFF FFFF ...
08x8000 0000 0 – 0x2F FFFF FFFF (2nd bank: if your memory exceeds 30G, place an additional 0G here)

10x0000 0000 0 – 88x0 FFFF FFFF ...
88x0000 0000 0 – 8x3F FFFF FFFF (32rd bank: if your memory exceeds 32G, place the rest up to <>G here)

이러한 경우 최대 3개의 뱅크가 존재하는 것일까요?

만일 4G 메모리를 모두 꽂아 64G를 채웠습니다. 이 때에는 3개의 뱅크가 맞죠.
그런데 만일 두 번째 뱅크 중간의 위치에 있는 4GB DRAM을 제거하였습니다.
그럼 몇 개의 뱅크일까요? 따라서 이러한 것을 하드웨어와 연결하여 생각하지 마시고,
커널 입장에서 물리 메모리가 한꺼번에 연달아 배치가 불가능하면 각각을 뱅크라고 생각하시면됩니다.

– – –

이번에는 섹션에 대해 말씀드립니다.
일단 arm 아키텍처에서 언급하는 섹션 매핑(1M)를 언급하는 것은 아닙니다.
동일한 용어를 사용하지만 리눅스 커널의 sparsemem에서 사용하는 섹션은 다른 것임을 먼저 말씀드립니다.

섹션은 hotplug를 고려하여 메모리를 추가 및 삭제하거나,
SOC 디자인 시 메모리의 배치가 어느 일정 규칙에 의해 배치되는 것을 고려하여 커널 컴파일 시 사이즈를 결정할 수 있습니다.
물론 커널 파라메터나 CONFIG로 시작하는 커널 옵션으로 설정할 수 없고, 보통 SoC 개발자가 결정합니다.
arm64의 경우 SECTION_SIZE_BITS=30을 사용하여 1GB를 디폴트로 사용합니다.
이 사이즈는 작으면 수십M에서 큰 경우 수G 단위로 지정하여 사용하는데 하드웨어적으로 물리메모리가 추가/제거될 수 있는 용량을 선택합니다.
예를 들어 내 서버가 최소 메모리의 증설이 512MB 단위이면 섹션 크기를 512MB로 설정합니다.
아직 ARM 시스템에서 하드웨어적으로 메모리를 핫플러그 하거나 핫리무브하는 경우는 없지만 다른 아키텍처들은 가능한 서버들이 있습니다.
또한 소프트웨어적으로는 하이퍼 바이저를 사용하여 리눅스 커널을 운영하는 경우 핫플러그 메모리 증설 단위를 128MB 단위로 운영하는 것이 효과적이다라고 판단하면 섹션 크기를 128M로 할 수 있습니다. (자주 사용되는 단위는 128MB, 256MB, 512MB, 1G, 4G 단위를 사용합니다)

응답 (/MM-1/?REPLYTOCOM=173050#RESPOND)

이승학
2017-10-12 19:05 (http://jake.dothome.co.kr/mm-1/#comment-173071)

빠른 답변 감사합니다. 도움이 많이 되었습니다.

응답 (/MM-1/?REPLYTOCOM=173071#RESPOND)

댓글 남기기

이메일은 공개되지 않습니다. 필수 입력창은 * 로 표시되어 있습니다

댓글

이름 *

이메일 *

웹사이트

댓글 작성

bootmem_init() (http://jake.dothome.co.kr/bootmem_init/)

zone_sizes_init() (http://jake.dothome.co.kr/free_area_init_node/)

문c 블로그 (2015 ~ 2024)