NUMA -3- (Memory policy)

2019-07-25 (http://jake.dothome.co.kr/numa-3-memory-policy/) Moon Young-il (http://jake.dothome.co.kr/author/admin/) Linux Kernel (http://jake.dothome.co.kr/category/linux/)

<kernel v5.0>

NUMA -3- (Memory policy)

Memory Policy

There are the following types of NUMA memory policies:

- MPOL_DEFAULT
  - This mode is only used inside the Memory Policy API. NULL and fallback internally to use the system's default memory policy.
- MPOL_PREFERRED
  - Specify and assign one preferred node. If you run out of memory, you can allocate it on another node.
  - However, if it is used with the MPOL_F_LOCAL flag, the preferred function is ignored and the local node allocation is given priority.
- MPOL_BIND
  - Allocate memory only on the specified bind nodes. If you run out of memory, you can't allocate it on other nodes.
- MPOL_INTERLEAVE
  - Round Robin at the specified interleaved nodes. If you run out of memory, you can allocate it on another node.
- MPOL_LOCAL
  - Change the mode to MPOL_PREFERRED, and use it with the MPOL_F_LOCAL flag to give preference to local nodes.

Memory Policy Flags

Use the following various memory policy-related flags:

- Flags used in set_mempolicy()
  - MPOL_F_STATIC_NODES(0x8000)
    - Specifying a static node
  - MPOL_F_RELATIVE_NODES(0x4000)
    - Specifying Relative Nodes
- Flags used in get_mempolicy()

  - MPOL_F_NODE(1)
    - Returning the next IL node instead of the node masque
  - MPOL_F_ADDR(2)
    - Search for VMA by address
  - MPOL_F_MEMS_ALLOWED(4)
    - return allowed memories
- Flags used in mbind()
  - MPOL_MF_STRICT(1)
  - MPOL_MF_MOVE(2)
  - MPOL_MF_MOVE_ALL(4)
  - MPOL_MF_LAZY(8)
  - MPOL_MF_INTERNAL(16)
- Internal flags used with mods
  - MPOL_F_SHARED(1)
    - Sharing Policies
  - MPOL_F_LOCAL(2)
    - Preferred Local Node Assignment
  - MPOL_F_MOF(8)
    - Migrate on fault
  - MPOL_F_MORON(16)
    - Migrate On protnone Reference On Node

The following shows that 20 CPU cores are used on each of the two nodes.

```
$ numactl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 20 21 22 23 24 25 26 27 28 29
node 0 size: 32654 MB
node 0 free: 18259 MB
node 1 cpus: 10 11 12 13 14 15 16 17 18 19 30 31 32 33 34 35 36 37 38 39
node 1 size: 32768 MB
node 1 free: 15491 MB
node distances:
node   0   1
  0:  10  21
  1:  21  10
```

The following shows how the Numa node policy uses default.

```
$ numactl --show
policy: default
preferred node: current
physcpubind: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
cpubind: 0 1
nodebind: 0 1
membind: 0 1
```

NUMA Memory Policy

default_policy Global Objects

mm/mempolicy.c

```
/*
 * run-time system-wide default policy => local allocation
 */
static struct mempolicy default_policy = {
        .refcnt = ATOMIC_INIT(1), /* never free it */
        .mode = MPOL_PREFERRED,
        .flags = MPOL_F_LOCAL,
};
```

Specify the memory policy to prioritize local nodes.

Knowing the memory policy for a task

get_task_policy()

mm/mempolicy.c

```
struct mempolicy *get_task_policy(struct task_struct *p)
{
        struct mempolicy *pol = p->mempolicy;
        int node;

        if (pol)
                return pol;

        node = numa_node_id();
        if (node != NUMA_NO_NODE) {
                pol = &preferred_node_policy[node];
                /* preferred_node_policy is not initialised early in boot */
                if (pol->mode)
                        return pol;
        }

        return &default_policy;
}
```

Returns the memory policy of the current task according to the following case:

- If the task has a policy specified, then -> 1) The memory policy specified in the task
- If the specified node exists, > 2) Use the specified node-first memory policy
- -> on request without a specified node 3) System Default Memory Policy (Local Node Preferred)

policy_node()

mm/mempolicy.c

```
/* Return the node id preferred by the given mempolicy, or the given id */
static int policy_node(gfp_t gfp, struct mempolicy *policy,
                                                                int nd)
{
        if (policy->mode == MPOL_PREFERRED && !(policy->flags & MPOL_F_LOCAL))
                nd = policy->v.preferred_node;
        else {
                /*
                 * __GFP_THISNODE shouldn't even be used with the bind policy
                 * because we might easily break the expectation to stay on the
                 * requested node and not break the policy.
                 */
                WARN_ON_ONCE(policy->mode == MPOL_BIND && (gfp & __GFP_THISNODE));
        }

        return nd;
}
```

If the memory policy is in Preferred mode and a Preferred node is specified, it returns the node number. If not, it returns an input argument @nd.

- If preferred requests to use a local node, it will simply return @nd.

policy_nodemask()

mm/mempolicy.c

```
/*
 * Return a nodemask representing a mempolicy for filtering nodes for
 * page allocation
 */
```

```
static nodemask_t *policy_nodemask(gfp_t gfp, struct mempolicy *policy)
{
        /* Lower zones don't get a nodemask applied for MPOL_BIND */
        if (unlikely(policy->mode == MPOL_BIND) &&
                        apply_policy_zone(policy, gfp_zone(gfp)) &&
                        cpuset_nodemask_valid_mems_allowed(&policy->v.nodes))
                return &policy->v.nodes;

        return NULL;
}
```

Interleaved Memory Policy

interleave_nodes()

mm/mempolicy.c

```
/* Do dynamic interleaving for a process */
static unsigned interleave_nodes(struct mempolicy *policy)
{
        unsigned nid, next;
        struct task_struct *me = current;

        nid = me->il_next;
        next = next_node(nid, policy->v.nodes);
        if (next >= MAX_NUMNODES)
                next = first_node(policy->v.nodes);
        if (next < MAX_NUMNODES)
                me->il_next = next;
        return nid;
}
```

Traversal the interleaved nodes and returns the node number.

- currnet->il_next In the meantime, keep track of the node you want to assign next in current->il_next.
- In il_next, nodes to be used when the memory policy is set to MPOL_INTERLEAVE are assigned by the interleave (round robin) method.
    - next_node()
        - Finds the next node of the specified node for the node bitmap. If it can't find it, it returns a MAX_NUMNODES.
    - first_node()
        - Node: Knows the node located at the beginning of the bitmap.

alloc_page_interleave()

mm/mempolicy.c

```
/* Allocate a page in interleaved policy.
   Own path because it needs to do special accounting. */
```

```
01 static struct page *alloc_page_interleave(gfp_t gfp, unsigned order,
02                                         unsigned nid)
03 {
04         struct page *page;
05 
06         page = __alloc_pages(gfp, order, nid);
07         /* skip NUMA_INTERLEAVE_HIT counter update if numa stats is disabled */
08         if (!static_branch_likely(&vm_numa_stat_key))
09                 return page;
10         if (page && page_to_nid(page) == nid) {
11                 preempt_disable();
12                 __inc_numa_state(page_zone(page), NUMA_INTERLEAVE_HIT);
13                 preempt_enable();
14         }
15         return page;
16 }
```

The interleave memory policy allocates 2^order pages of contiguous physical memory.

- The gfp flag requested in line 6 of the code, the node allocates 2^order pages of contiguous physical memory.
    - Depending on the gfp flag, you can either get a full node zonelist of node_zonelist[0], or a node_zonelist[1] containing only the specified node zones.
- If you are not using the NUMA statistic in line 8~9 of the code, skip it.
- Update the NUMA statistics on lines 10~14 of the code.
    - Only if the page is assigned from the requested node will the NUMA_INTERLEAVE_HIT stat of the zone be incremented.

Main Structure

mempolicy struct

include/linux/mempolicy.h

```
/*
 * Describe a memory policy.
 *
 * A mempolicy can be either associated with a process or with a VMA.
 * For VMA related allocations the VMA policy is preferred, otherwise
 * the process policy is used. Interrupts ignore the memory policy
 * of the current process.
 *
 * Locking policy for interlave:
 * In process context there is no locking because only the process accesses
 * its own state. All vma manipulation is somewhat protected by a down_read on
 * mmap_sem.
 *
 * Freeing policy:
 * Mempolicy objects are reference counted.  A mempolicy will be freed when
 * mpol_put() decrements the reference count to zero.
 *
 * Duplicating policy objects:
 * mpol_dup() allocates a new mempolicy and copies the specified mempolicy
 * to the new storage.  The reference count of the new object is initialized
 * to 1, representing the caller of mpol_dup().
 */
```

```
struct mempolicy {
        atomic_t refcnt;
        unsigned short mode;    /* See MPOL_* above */
        unsigned short flags;   /* See set_mempolicy() MPOL_F_* above */
        union {
                short            preferred_node; /* preferred */
                nodemask_t       nodes;         /* interleave/bind */
                /* undefined for default */
        } v;
        union {
                nodemask_t cpuset_mems_allowed; /* relative to these nodes */
                nodemask_t user_nodemask;       /* nodemask passed by user */
        } w;
};
```

consultation

- NUMA with Linux (https://lunatine.net/2016/07/14/numa-with-linux/) | Lunatine's Box
- Local and Remote Memory: Memory in a Linux/NUMA System | Christoph Lameter – pdf 다운로드 (http://gentwo.org/christoph/Presentations-2000-2009/2006-OLS-Local_and_Remote_Memory.pdf)
- NUMA Best Practices for Dell PowerEdge 12th Generation Servers | Dell – pdf 다운로드 (https://lafibre.info/images/materiel/201212_dell_bios_tuning_for_performance.pdf)
- What is Linux Memory Policy? (https://www.kernel.org/doc/Documentation/vm/numa_memory_policy.txt) | kernel.org
- NUMA API for Linux (https://lwn.net/Articles/79100/) | LWN.net
- numa – overview of Non-Uniform Memory Architecture (http://man7.org/linux/man-pages/man7/numa.7.html) | man7.org

- NUMA (Non-Uniform Memory Access): An Overview (http://queue.acm.org/detail.cfm?id=2513149) | queue.acm.org
- Red Hat Enterprise Linux Non-Uniform Memory Access support for HP ProLiant servers | HP – pdf 다운로드 (https://h50146.www5.hpe.com/products/software/oe/linux/mainstream/support/whitepaper/pdfs/a00039147enw.pdf)

LEAVE A COMMENT

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

Comments

name *

email *

Website

WRITE A COMMENT

< Debug Memory-4- (Idle Page Trace) (http://jake.dothome.co.kr/debug-mem-4/)

kernel/head. S – ARM64 (old for v5.0) > (http://jake.dothome.co.kr/head-64/)

Munc Blog (2015 ~ 2024)