Subject: Re: [PATCH][for -mm] per-zone and reclaim enhancements for memory controller take 3 [3/10] per-zone
Posted by Lee Schermerhorn on Wed, 28 Nov 2007 21:19:59 GMT
View Forum Message <> Reply to Message

Just a "heads up": This patch is the apparent cause of a boot time panic--null pointer deref--on my numa platform. See below.

On Tue, 2007-11-27 at 12:00 +0900, KAMEZAWA Hiroyuki wrote:
> Counting active/inactive per-zone in memory controller.
>
> This patch adds per-zone status in memory cgroup.
> These values are often read (as per-zone value) by page reclaiming.
>
> In current design, per-zone stat is just a unsigned long value and
> not an atomic value because they are modified only under lru_lock.
> (So, atomic_ops is not necessary.)
>
> This patch adds ACTIVE and INACTIVE per-zone status values.
>
> For handling per-zone status, this patch adds
> struct mem_cgroup_per_zone {
> ...
> }
> and some helper functions. This will be useful to add per-zone objects
> in mem_cgroup.
>
> This patch turns memory controller's early_init to be 0 for calling
> kmalloc() in initialization.
>
> Changelog V2 -> V3
> - fixed comments.
>
> Changelog V1 -> V2
> - added mem_cgroup_per_zone struct.
> This will help following patches to implement per-zone objects and
> pack them into a struct.
> - added __mem_cgroup_add_list() and __mem_cgroup_remove_list()
> - fixed page migration handling.
> - renamed zstat to info (per-zone-info)
> This will be place for per-zone information(lru, lock, ..)
> - use page_cgroup_nid()/zid() funcs.
>
> Acked-by: Balbir Singh <balbir@linux.vnet.ibm.com>
> Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
>
>
> mm/memcontrol.c | 164

++---
> 1 file changed, 157 insertions(+), 7 deletions(-)
>
> Index: linux-2.6.24-rc3-mm1/mm/memcontrol.c
> ===
> --- linux-2.6.24-rc3-mm1.orig/mm/memcontrol.c 2007-11-26 16:39:00.000000000 +0900
> +++ linux-2.6.24-rc3-mm1/mm/memcontrol.c 2007-11-26 16:39:02.000000000 +0900
> @@ -78,6 +78,31 @@
<snip>
>
> +static int alloc_mem_cgroup_per_zone_info(struct mem_cgroup *mem, int node)
> +{
> + struct mem_cgroup_per_node *pn;
> +
> + pn = kmalloc_node(sizeof(*pn), GFP_KERNEL, node);
> + if (!pn)
> + return 1;
> + mem->info.nodeinfo[node] = pn;
> + memset(pn, 0, sizeof(*pn));
> + return 0;
> +}
> +
> static struct mem_cgroup init_mem_cgroup;
>
> static struct cgroup_subsys_state *
> mem_cgroup_create(struct cgroup_subsys *ss, struct cgroup *cont)
> {
>   struct mem_cgroup *mem;
> + int node;
>
>   if (unlikely((cont->parent) == NULL)) {
>   mem = &init_mem_cgroup;
> @@ -907,7 +1039,19 @@
>   INIT_LIST_HEAD(&mem->inactive_list);
>   spin_lock_init(&mem->lru_lock);
>   mem->control_type = MEM_CGROUP_TYPE_ALL;
> + memset(&mem->info, 0, sizeof(mem->info));
> +
> + for_each_node_state(node, N_POSSIBLE)
> + if (alloc_mem_cgroup_per_zone_info(mem, node))
> +   goto free_out;
> +

As soon as this loop hits the first non-existent node on my platform, I
get a NULL pointer deref down in __alloc_pages. Stack trace below.

Perhaps N_POSSIBLE should be N_HIGH_MEMORY? That would require handling
of memory/node hotplug for each memory control group, right? But, I'm

going to try N_HIGH_MEMORY as a work around.

Lee
> return &mem->css;
> +free_out:
> + for_each_node_state(node, N_POSSIBLE)
> + kfree(mem->info.nodeinfo[node]);
> + if (cont->parent != NULL)
> + kfree(mem);
> + return NULL;
> }
>
> static void mem_cgroup_pre_destroy(struct cgroup_subsys *ss,
> @@ -920,6 +1064,12 @@
> static void mem_cgroup_destroy(struct cgroup_subsys *ss,
>   struct cgroup *cont)
> {
> + int node;
> + struct mem_cgroup *mem = mem_cgroup_from_cont(cont);
> +
> + for_each_node_state(node, N_POSSIBLE)
> + kfree(mem->info.nodeinfo[node]);
> +
> kfree(mem_cgroup_from_cont(cont));
> }
>
> @@ -972,5 +1122,5 @@
> .destroy = mem_cgroup_destroy,
> .populate = mem_cgroup_populate,
> .attach = mem_cgroup_move_task,
> - .early_init = 1,
> + .early_init = 0,
> };

Initializing cgroup subsys memory
Unable to handle kernel NULL pointer dereference (address 0000000000003c80)
swapper[0]: Oops 11012296146944 [1]
Modules linked in:

Pid: 0, CPU 0, comm: swapper
psr : 00001210084a6010 ifs : 8000000000000b1a ip : [<a000000100132e11>]   Not tainted
ip is at __alloc_pages+0x31/0x6e0
unat: 0000000000000000 pfs : 000000000000060f rsc : 0000000000000003
rnat: a0000001009db3b8 bsps: a0000001009e0490 pr : 656960155aa65659
ldrs: 0000000000000000 ccv : 0000000000000000 fpsr: 0009804c8a70433f
csd : 0000000000000000 ssd : 0000000000000000
b0 : a000000100187370 b6 : a000000100194440 b7 : a00000010086d560
f6 : 1003e0000000000000000 f7 : 1003e0000000000000055

f8 : 1003e00000000000000c0 f9 : 1003e0000000000003fc0
f10 : 1003e00000000000000c0 f11 : 1003e0000000000000055
r1 : a000000100bc0f10 r2 : fffffffffe00006 r3 : 0000000000020000
r8 : 0000000000071ef0 r9 : 0000000000000005 r10 : e00007002034d588
r11 : e00007002034d580 r12 : a0000001008e3df0 r13 : a0000001008dc000
r14 : 0000000000000001 r15 : e00007002034d5b0 r16 : 0000000000001e78
r17 : ffffffffffff04e0 r18 : 0000000000100002 r19 : 0000000000000000
r20 : 0000000000100002 r21 : 00000000000003cf r22 : 000000000000000f
r23 : 00000000000003c0 r24 : 0000000000000010 r25 : 0000000000000001
r26 : a0000001008e3e20 r27 : 0000000000000000 r28 : e0000701813dc088
r29 : e0000701813dc080 r30 : 0000000000000000 r31 : a000000100918ea8

Call Trace:
 [<a000000100014de0>] show_stack+0x80/0xa0
                        sp=a0000001008e39c0 bsp=a0000001008dd1b0
 [<a000000100015a70>] show_regs+0x870/0x8a0
                        sp=a0000001008e3b90 bsp=a0000001008dd158
 [<a00000010003d130>] die+0x190/0x300
                        sp=a0000001008e3b90 bsp=a0000001008dd110
 [<a000000100071b80>] ia64_do_page_fault+0x8e0/0xa20
                        sp=a0000001008e3b90 bsp=a0000001008dd0b8
 [<a00000010000b5c0>] ia64_leave_kernel+0x0/0x270
                        sp=a0000001008e3c20 bsp=a0000001008dd0b8
 [<a000000100132e10>] __alloc_pages+0x30/0x6e0
                        sp=a0000001008e3df0 bsp=a0000001008dcfe0
 [<a000000100187370>] new_slab+0x610/0x6c0
                        sp=a0000001008e3e00 bsp=a0000001008dcf80
 [<a000000100187470>] get_new_slab+0x50/0x200
                        sp=a0000001008e3e00 bsp=a0000001008dcf48
 [<a000000100187900>] __slab_alloc+0x2e0/0x4e0
                        sp=a0000001008e3e00 bsp=a0000001008dcf00
 [<a000000100187c80>] kmem_cache_alloc_node+0x180/0x200
                        sp=a0000001008e3e10 bsp=a0000001008dcec0
 [<a0000001001945a0>] mem_cgroup_create+0x160/0x400
                        sp=a0000001008e3e10 bsp=a0000001008dce78
 [<a0000001000f0940>] cgroup_init_subsys+0xa0/0x400
                        sp=a0000001008e3e20 bsp=a0000001008dce28
 [<a0000001008521f0>] cgroup_init+0x90/0x160
                        sp=a0000001008e3e20 bsp=a0000001008dce00
 [<a000000100831960>] start_kernel+0x700/0x820
                        sp=a0000001008e3e20 bsp=a0000001008dcd80