

Subject: [RFC][PATCH] memory controller per zone patches take 2 [7/10] calculate reclaim scan number for memo

Posted by [KAMEZAWA Hiroyuki](#) on Fri, 16 Nov 2007 10:24:02 GMT

[View Forum Message](#) <> [Reply to Message](#)

Define function for calculating the number of scan target on each Zone/LRU.

Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>

```
include/linux/memcontrol.h | 15 ++++++
mm/memcontrol.c            | 33 ++++++
2 files changed, 48 insertions(+)
```

Index: linux-2.6.24-rc2-mm1/include/linux/memcontrol.h

=====

--- linux-2.6.24-rc2-mm1.orig/include/linux/memcontrol.h

+++ linux-2.6.24-rc2-mm1/include/linux/memcontrol.h

```
@@ -73,6 +73,10 @@ extern void mem_cgroup_note_reclaim_prio
extern void mem_cgroup_record_reclaim_priority(struct mem_cgroup *mem,
int priority);
```

```
+extern long mem_cgroup_calc_reclaim_active(struct mem_cgroup *mem,
+ struct zone *zone, int priority);
+extern long mem_cgroup_calc_reclaim_inactive(struct mem_cgroup *mem,
+ struct zone *zone, int priority);
```

```
#else /* CONFIG_CGROUP_MEM_CONT */
static inline void mm_init_cgroup(struct mm_struct *mm,
@@ -173,6 +177,17 @@ static inline void mem_cgroup_record_rec
return 0;
}
```

```
+static inline long mem_cgroup_calc_reclaim_active(struct mem_cgroup *mem,
+ struct zone *zone, int priority)
+{
+ return 0;
+}
+
+static inline long mem_cgroup_calc_reclaim_inactive(struct mem_cgroup *mem,
+ struct zone *zone, int priority)
+{
+ return 0;
+}
#endif /* CONFIG_CGROUP_MEM_CONT */
```

```
#endif /* _LINUX_MEMCONTROL_H */
```

Index: linux-2.6.24-rc2-mm1/mm/memcontrol.c

=====

```

--- linux-2.6.24-rc2-mm1.orig/mm/memcontrol.c
+++ linux-2.6.24-rc2-mm1/mm/memcontrol.c
@@ -472,6 +472,39 @@ void mem_cgroup_record_reclaim_priority(
    mem->prev_priority = priority;
}

+/*
+ * Calculate # of pages to be scanned in this priority/zone.
+ * See also vmscan.c
+ *
+ * priority starts from "DEF_PRIORITY" and decremented in each loop.
+ * (see include/linux/mmzone.h)
+ */
+
+long mem_cgroup_calc_reclaim_active(struct mem_cgroup *mem,
+    struct zone *zone, int priority)
+{
+    s64 nr_active;
+    int nid = zone->zone_pgdat->node_id;
+    int zid = zone_idx(zone);
+    struct mem_cgroup_per_zone *mz = mem_cgroup_zoneinfo(mem, nid, zid);
+
+    nr_active = MEM_CGROUP_ZSTAT(mz, MEM_CGROUP_ZSTAT_ACTIVE);
+    return (long)(nr_active >> priority);
+}
+
+long mem_cgroup_calc_reclaim_inactive(struct mem_cgroup *mem,
+    struct zone *zone, int priority)
+{
+    u64 nr_inactive;
+    int nid = zone->zone_pgdat->node_id;
+    int zid = zone_idx(zone);
+    struct mem_cgroup_per_zone *mz = mem_cgroup_zoneinfo(mem, nid, zid);
+
+    nr_inactive = MEM_CGROUP_ZSTAT(mz, MEM_CGROUP_ZSTAT_INACTIVE);
+
+    return (long)(nr_inactive >> priority);
+}
+
+unsigned long mem_cgroup_isolate_pages(unsigned long nr_to_scan,
+    struct list_head *dst,
+    unsigned long *scanned, int order,

```

Containers mailing list
Containers@lists.linux-foundation.org
<https://lists.linux-foundation.org/mailman/listinfo/containers>
