Subject: [PATCH 1/4] swapcgroup: add cgroup files
Posted by Daisuke Nishimura on Thu, 22 May 2008 06:17:26 GMT
View Forum Message <> Reply to Message

This patch add cgroup files(and a member to struct mem_cgroup)
for swapcgroup.

The files to be added are:

- memory.swap_usage_in_bytes
- memory.swap_max_usage_in_bytes
- memory.swap_limit_in_bytes
- memory.swap_failcnt

The meaning of those files are the same as memory cgroup.

Signed-off-by: Daisuke Nishimura <nishimura@mxp.nes.nec.co.jp>

```
---
 init/Kconfig     |    7 +++++
 mm/memcontrol.c |   67 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 2 files changed, 74 insertions(+), 0 deletions(-)

diff --git a/init/Kconfig b/init/Kconfig
index 6135d07..aabbb83 100644
--- a/init/Kconfig
+++ b/init/Kconfig
@@ -407,6 +407,13 @@ config CGROUP_MEM_RES_CTLR
    This config option also selects MM_OWNER config option, which
    could in turn add some fork/exit overhead.

+config CGROUP_SWAP_RES_CTLR
+ bool "Swap Resource Controller for Control Groups"
+ depends on CGROUP_MEM_RES_CTLR && SWAP
+ help
+   Provides a swap resource controller that manages and limits swap usage.
+   Implemented as a add-on to Memory Resource Controller.
+
 config SYSFS_DEPRECATED
  bool

diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index a96577f..a837215 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -141,6 +141,12 @@ struct mem_cgroup {
   * statistics.
   */
  struct mem_cgroup_stat stat;
+#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
+ /*
+  * the counter to account for swap usage
+  */
+ struct res_counter swap_res;
+#endif
 };
 static struct mem_cgroup init_mem_cgroup;

@@ -960,6 +966,39 @@ static int mem_control_stat_show(struct cgroup *cont, struct cftype *cft,
  return 0;
 }

+#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
+static u64 swap_cgroup_read(struct cgroup *cont, struct cftype *cft)
+{
+ return res_counter_read_u64(&mem_cgroup_from_cont(cont)->swap_res,
+       cft->private);
+}
+
+static ssize_t swap_cgroup_write(struct cgroup *cont, struct cftype *cft,
+    struct file *file, const char __user *userbuf,
+    size_t nbytes, loff_t *ppos)
+{
+ return res_counter_write(&mem_cgroup_from_cont(cont)->swap_res,
+    cft->private, userbuf, nbytes, ppos,
+    mem_cgroup_write_strategy);
+}
+
+static int swap_cgroup_reset(struct cgroup *cont, unsigned int event)
+{
+ struct mem_cgroup *mem;
+
+ mem = mem_cgroup_from_cont(cont);
+ switch (event) {
+ case RES_MAX_USAGE:
+  res_counter_reset_max(&mem->swap_res);
+  break;
+ case RES_FAILCNT:
+  res_counter_reset_failcnt(&mem->swap_res);
+  break;
+ }
+ return 0;
+}
+#endif
+
 static struct cftype mem_cgroup_files[] = {
  {
   .name = "usage_in_bytes",
@@ -992,6 +1031,31 @@ static struct cftype mem_cgroup_files[] = {
   .name = "stat",
   .read_map = mem_control_stat_show,
  },
+#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
+ {
+  .name = "swap_usage_in_bytes",
+  .private = RES_USAGE,
+  .read_u64 = swap_cgroup_read,
+ },
+ {
+  .name = "swap_max_usage_in_bytes",
+  .private = RES_MAX_USAGE,
+  .trigger = swap_cgroup_reset,
+  .read_u64 = swap_cgroup_read,
+ },
+ {
+  .name = "swap_limit_in_bytes",
+  .private = RES_LIMIT,
+  .write = swap_cgroup_write,
+  .read_u64 = swap_cgroup_read,
+ },
+ {
+  .name = "swap_failcnt",
+  .private = RES_FAILCNT,
+  .trigger = swap_cgroup_reset,
+  .read_u64 = swap_cgroup_read,
+ },
+#endif
 };

 static int alloc_mem_cgroup_per_zone_info(struct mem_cgroup *mem, int node)
@@ -1069,6 +1133,9 @@ mem_cgroup_create(struct cgroup_subsys *ss, struct cgroup *cont)
  }

  res_counter_init(&mem->res);
+#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
+ res_counter_init(&mem->swap_res);
+#endif

  for_each_node_state(node, N_POSSIBLE)
   if (alloc_mem_cgroup_per_zone_info(mem, node))
```