Subject: [PATCH v8 8/9] Display current tcp failcnt in kmem cgroup
Posted by Glauber Costa on Mon, 05 Dec 2011 21:35:02 GMT
View Forum Message <> Reply to Message

This patch introduces kmem.tcp.failcnt file, living in the
kmem_cgroup filesystem. Following the pattern in the other
memcg resources, this files keeps a counter of how many times
allocation failed due to limits being hit in this cgroup.
The root cgroup will always show a failcnt of 0.

Signed-off-by: Glauber Costa <glommer@parallels.com>
Reviewed-by: Hiroyouki Kamezawa <kamezawa.hiroyu@jp.fujitsu.com>
CC: David S. Miller <davem@davemloft.net>
CC: Eric W. Biederman <ebiederm@xmission.com>
---
 net/ipv4/tcp_memcontrol.c | 31 +++++++++++++++++++++++++++++++
 1 files changed, 31 insertions(+), 0 deletions(-)

```
diff --git a/net/ipv4/tcp_memcontrol.c b/net/ipv4/tcp_memcontrol.c
index 9481f23..d438fba 100644
--- a/net/ipv4/tcp_memcontrol.c
+++ b/net/ipv4/tcp_memcontrol.c
@@ -9,6 +9,7 @@
 static u64 tcp_cgroup_read(struct cgroup *cont, struct cftype *cft);
 static int tcp_cgroup_write(struct cgroup *cont, struct cftype *cft,
        const char *buffer);
+static int tcp_cgroup_reset(struct cgroup *cont, unsigned int event);

 static struct cftype tcp_files[] = {
  {
@@ -22,6 +23,12 @@ static struct cftype tcp_files[] = {
   .read_u64 = tcp_cgroup_read,
   .private = RES_USAGE,
  },
+ {
+  .name = "kmem.tcp.failcnt",
+  .private = RES_FAILCNT,
+  .trigger = tcp_cgroup_reset,
+  .read_u64 = tcp_cgroup_read,
+ },
 };

 static inline struct tcp_memcontrol *tcp_from_cgproto(struct cg_proto *cg_proto)
@@ -197,12 +204,36 @@ static u64 tcp_cgroup_read(struct cgroup *cont, struct cftype *cft)
  case RES_USAGE:
   val = tcp_read_usage(memcg);
   break;
+ case RES_FAILCNT:
```

```
+  val = tcp_read_stat(memcg, RES_FAILCNT, 0);
+  break;
  default:
   BUG();
  }
  return val;
 }

+static int tcp_cgroup_reset(struct cgroup *cont, unsigned int event)
+{
+ struct mem_cgroup *memcg;
+ struct tcp_memcontrol *tcp;
+ struct cg_proto *cg_proto;
+
+ memcg = mem_cgroup_from_cont(cont);
+ cg_proto = tcp_prot.proto_cgroup(memcg);
+ if (!cg_proto)
+  return 0;
+ tcp = tcp_from_cgproto(cg_proto);
+
+ switch (event) {
+ case RES_FAILCNT:
+  res_counter_reset_failcnt(&tcp->tcp_memory_allocated);
+  break;
+ }
+
+ return 0;
+}
+
 unsigned long long tcp_max_memory(const struct mem_cgroup *memcg)
 {
  struct tcp_memcontrol *tcp;
--
1.7.6.4
```