Subject: [RFC] special case cpuacct cpuusage when cpu cgroup is comounted
Posted by Glauber Costa on Thu, 22 Dec 2011 09:06:22 GMT
View Forum Message <> Reply to Message

The cpuacct cgroup needs to do hierarchy walks in scheduler core paths
to update its cpuusage information. For the cfs case it is even worse, since
we already track this data in cfs_rq's exec_clock.

We can't really get rid of it in a clean way, but we can do a lot better
when cpu an cpuacct are mounted in the same hierarchy: in this case, we're
guaranteed to have the same set of tasks. There is then no need to walk
the cpuacct hierachy, since what we record for the task_group will yield
the same results.

For the rt tasks, we need to record tg's rt_rq total execution time
somewhere in the struct, since it was not provided before.

Please let me know what you guys think of this.

Signed-off-by: Glauber Costa <glommer@parallels.com>
CC: Paul Turner <pjt@google.com>
CC: Li Zefan <lizf@cn.fujitsu.com>
CC: Peter Zijlstra <a.p.zijlstra@chello.nl>
---
 kernel/sched/core.c | 198 ++---
 kernel/sched/rt.c | 6 ++
 kernel/sched/sched.h | 4 +
 3 files changed, 195 insertions(+), 13 deletions(-)

diff --git a/kernel/sched/core.c b/kernel/sched/core.c
index a28034d..d44ce99 100644
--- a/kernel/sched/core.c
+++ b/kernel/sched/core.c
@@ -6810,6 +6810,45 @@ int in_sched_functions(unsigned long addr)
 struct task_group root_task_group;
 #endif

+#if defined(CONFIG_CGROUP_SCHED) && defined(CONFIG_CGROUP_CPUACCT)
+struct jump_label_key cpu_cpuacct_comounted_branch;
+
+static void cpu_cpuacct_update_jump_labels(struct cgroup *root,
+   struct cgroupfs_root *sys_root)
+{
+ /* When we are adding a cgroup, we first bind it to the new root,
+ * then call bind().
+ * When deleting one, we first call bind(), then move it away.
+ *
+ * As a result, when the two roots are different, we can be sure that

```
+  * we're either adding the first one of them in a hierarchy, or we
+  * are deleting the last of them. In any of those cases, we don't need
+  * to mess with the jump labels. (first test)
+  *
+  * So if our root equals the root bind() were provided, that's because
+  * we are doing an addition (update root first, then call bind())
+  *
+  * Analog logic for removal
+  */
+ if (root_cpuacct.css.cgroup->root != root_task_group.css.cgroup->root)
+  return;
+
+ if (root->root == sys_root)
+  jump_label_inc(&cpu_cpuacct_comounted_branch);
+ else
+  jump_label_dec(&cpu_cpuacct_comounted_branch);
+
+}
+
+#define cpu_cpuacct_comounted static_branch(&cpu_cpuacct_comounted_branch)
+#elif defined(CONFIG_CGROUP_SCHED) || defined(CONFIG_CGROUP_CPUACCT)
+static void cpu_cpuacct_update_jump_labels(struct cgroup *cgrp,
+        struct cgroupfs_root *sys_root)
+{
+}
+#define cpu_cpuacct_comounted 0
+#endif
+
 DECLARE_PER_CPU(cpumask_var_t, load_balance_tmpmask);

 void __init sched_init(void)
@@ -7916,6 +7955,11 @@ static int cpu_cgroup_populate(struct cgroup_subsys *ss, struct
cgroup *cont)
  return cgroup_add_files(cont, ss, cpu_files, ARRAY_SIZE(cpu_files));
 }

+void cpu_cgroup_bind(struct cgroup_subsys *ss, struct cgroup *root)
+{
+ cpu_cpuacct_update_jump_labels(root, root_task_group.css.cgroup->root);
+}
+
 struct cgroup_subsys cpu_cgroup_subsys = {
  .name  = "cpu",
  .create  = cpu_cgroup_create,
@@ -7925,6 +7969,7 @@ struct cgroup_subsys cpu_cgroup_subsys = {
  .exit  = cpu_cgroup_exit,
  .populate = cpu_cgroup_populate,
  .subsys_id = cpu_cgroup_subsys_id,
```

```
+ .bind  = cpu_cgroup_bind,
  .early_init = 1,
 };

@@ -7981,8 +8026,102 @@ cpuacct_destroy(struct cgroup_subsys *ss, struct cgroup *cgrp)
  kfree(ca);
 }

-static u64 cpuacct_cpuusage_read(struct cpuacct *ca, int cpu)
+#ifdef CONFIG_CGROUP_SCHED
+#ifdef CONFIG_FAIR_GROUP_SCHED
+static struct cfs_rq *
+cpu_cgroup_cfs_rq(struct cgroup *cgrp, int cpu)
+{
+ struct task_group *tg = cgroup_tg(cgrp);
+
+ if (tg == &root_task_group)
+  return &cpu_rq(cpu)->cfs;
+
+ return tg->cfs_rq[cpu];
+}
+
+static void cpu_cgroup_update_cpuusage_cfs(struct cgroup *cgrp, int cpu)
+{
+ struct cfs_rq *cfs = cpu_cgroup_cfs_rq(cgrp, cpu);
+ cfs->prev_exec_clock = cfs->exec_clock;
+}
+static u64 cpu_cgroup_cpuusage_cfs(struct cgroup *cgrp, int cpu)
+{
+ struct cfs_rq *cfs = cpu_cgroup_cfs_rq(cgrp, cpu);
+ return cfs->exec_clock - cfs->prev_exec_clock;
+}
+#else
+static void cpu_cgroup_update_cpuusage_cfs(struct cgroup *cgrp, int cpu)
+{
+}
+
+static u64 cpu_cgroup_cpuusage_cfs(struct cgroup *cgrp, int cpu)
+{
+ return 0;
+}
+#endif
+
+#ifdef CONFIG_RT_GROUP_SCHED
+static struct rt_rq *
+cpu_cgroup_rt_rq(struct cgroup *cgrp, int cpu)
+{
+ struct task_group *tg = cgroup_tg(cgrp);
```

```
+ if (tg == &root_task_group)
+  return &cpu_rq(cpu)->rt;
+
+ return tg->rt_rq[cpu];
+
+}
+static void cpu_cgroup_update_cpuusage_rt(struct cgroup *cgrp, int cpu)
+{
+ struct rt_rq *rt = cpu_cgroup_rt_rq(cgrp, cpu);
+ rt->prev_exec_clock = rt->exec_clock;
+}
+
+static u64 cpu_cgroup_cpuusage_rt(struct cgroup *cgrp, int cpu)
+{
+ struct rt_rq *rt = cpu_cgroup_rt_rq(cgrp, cpu);
+ return rt->exec_clock - rt->prev_exec_clock;
+}
+#else
+static void cpu_cgroup_update_cpuusage_rt(struct cgroup *cgrp, int cpu)
+{
+}
+static u64 cpu_cgroup_cpuusage_rt(struct cgroup *cgrp, int cpu)
+{
+ return 0;
+}
+#endif
+
+static int cpu_cgroup_cpuusage_write(struct cgroup *cgrp, int cpu, u64 val)
 {
+ cpu_cgroup_update_cpuusage_cfs(cgrp, cpu);
+ cpu_cgroup_update_cpuusage_rt(cgrp, cpu);
+ return 0;
+}
+
+static u64 cpu_cgroup_cpuusage_read(struct cgroup *cgrp, int cpu)
+{
+ return cpu_cgroup_cpuusage_cfs(cgrp, cpu) +
+        cpu_cgroup_cpuusage_rt(cgrp, cpu);
+}
+
+#else
+static u64 cpu_cgroup_cpuusage_read(struct cgroup *cgrp, int i)
+{
+ BUG();
+ return 0;
+}
+
+static int cpu_cgroup_cpuusage_write(struct cgroup *cgrp, int cpu, u64 val)
```

```
+{
+ BUG();
+ return 0;
+}
+#endif /* CONFIG_CGROUP_SCHED */
+
+static u64 cpuacct_cpuusage_read(struct cgroup *cgrp, int cpu)
+{
+ struct cpuacct *ca = cgroup_ca(cgrp);
  u64 *cpuusage = per_cpu_ptr(ca->cpuusage, cpu);
  u64 data;

@@ -8000,8 +8139,9 @@ static u64 cpuacct_cpuusage_read(struct cpuacct *ca, int cpu)
  return data;
 }

-static void cpuacct_cpuusage_write(struct cpuacct *ca, int cpu, u64 val)
+static void cpuacct_cpuusage_write(struct cgroup *cgrp, int cpu, u64 val)
 {
+ struct cpuacct *ca = cgroup_ca(cgrp);
  u64 *cpuusage = per_cpu_ptr(ca->cpuusage, cpu);

 #ifndef CONFIG_64BIT
@@ -8016,15 +8156,22 @@ static void cpuacct_cpuusage_write(struct cpuacct *ca, int cpu, u64
val)
 #endif
 }

+static u64 cpuusage_read_percpu(struct cgroup *cgrp, int cpu)
+{
+ if (cpu_cpuacct_comounted)
+  return cpu_cgroup_cpuusage_read(cgrp, cpu);
+ else
+  return cpuacct_cpuusage_read(cgrp, cpu);
+}
+
 /* return total cpu usage (in nanoseconds) of a group */
 static u64 cpuusage_read(struct cgroup *cgrp, struct cftype *cft)
 {
- struct cpuacct *ca = cgroup_ca(cgrp);
  u64 totalcpuusage = 0;
  int i;

  for_each_present_cpu(i)
-  totalcpuusage += cpuacct_cpuusage_read(ca, i);
+  totalcpuusage += cpuusage_read_percpu(cgrp, i);

  return totalcpuusage;
```

```
 }
@@ -8032,7 +8179,6 @@ static u64 cpuusage_read(struct cgroup *cgrp, struct cftype *cft)
 static int cpuusage_write(struct cgroup *cgrp, struct cftype *cftype,
         u64 reset)
 {
- struct cpuacct *ca = cgroup_ca(cgrp);
   int err = 0;
   int i;

@@ -8041,8 +8187,12 @@ static int cpuusage_write(struct cgroup *cgrp, struct cftype *cftype,
    goto out;
   }

- for_each_present_cpu(i)
-   cpuacct_cpuusage_write(ca, i, 0);
+ for_each_present_cpu(i) {
+   if (cpu_cpuacct_comounted)
+     cpu_cgroup_cpuusage_write(cgrp, i, 0);
+   else
+     cpuacct_cpuusage_write(cgrp, i, 0);
+ }

 out:
   return err;
@@ -8051,12 +8201,11 @@ out:
 static int cpuacct_percpu_seq_read(struct cgroup *cgroup, struct cftype *cft,
         struct seq_file *m)
 {
- struct cpuacct *ca = cgroup_ca(cgroup);
   u64 percpu;
   int i;

   for_each_present_cpu(i) {
-    percpu = cpuacct_cpuusage_read(ca, i);
+    percpu = cpuusage_read_percpu(cgroup, i);
     seq_printf(m, "%llu ", (unsigned long long) percpu);
   }
   seq_printf(m, "\n");
@@ -8123,14 +8272,11 @@ static int cpuacct_populate(struct cgroup_subsys *ss, struct cgroup
*cgrp)
   *
   * called with rq->lock held.
   */
-void cpuacct_charge(struct task_struct *tsk, u64 cputime)
+static void __cpuacct_charge(struct task_struct *tsk, u64 cputime)
 {
   struct cpuacct *ca;
   int cpu;
```

```
- if (unlikely(!cpuacct_subsys.active))
-  return;
-
  cpu = task_cpu(tsk);

  rcu_read_lock();
@@ -8145,11 +8291,37 @@ void cpuacct_charge(struct task_struct *tsk, u64 cputime)
  rcu_read_unlock();
 }

+void cpuacct_charge(struct task_struct *tsk, u64 cputime)
+{
+ if (unlikely(!cpuacct_subsys.active))
+  return;
+
+/*
+ * If schedstats are not enabled, we don't accumulate task_group
+ * execution time in exec_clock. Therefore, we need to walk the
+ * structure all over again here.
+ *
+ * If we're comounted, we'll just use the same tg pointers already
+ * in the cache, since we now use cgrp's parent as cpuacct parent
+ * pointer
+ */
+#ifdef CONFIG_SCHEDSTATS
+ if (!cpu_cpuacct_comounted)
+#endif
+  __cpuacct_charge(tsk, cputime);
+}
+
+void cpuacct_bind(struct cgroup_subsys *ss, struct cgroup *root)
+{
+ cpu_cpuacct_update_jump_labels(root, root_cpuacct.css.cgroup->root);
+}
+
 struct cgroup_subsys cpuacct_subsys = {
  .name = "cpuacct",
  .create = cpuacct_create,
  .destroy = cpuacct_destroy,
  .populate = cpuacct_populate,
  .subsys_id = cpuacct_subsys_id,
+ .bind = cpuacct_bind,
 };
 #endif /* CONFIG_CGROUP_CPUACCT */
diff --git a/kernel/sched/rt.c b/kernel/sched/rt.c
index 3640ebb..172833d 100644
--- a/kernel/sched/rt.c
```

```
+++ b/kernel/sched/rt.c
@@ -894,6 +895,11 @@ static void update_curr_rt(struct rq *rq)

  sched_rt_avg_update(rq, delta_exec);

+ for_each_sched_rt_entity(rt_se) {
+  rt_rq = rt_rq_of_se(rt_se);
+  schedstat_add(rt_rq, exec_clock, delta_exec);
+ }
+
  if (!rt_bandwidth_enabled())
   return;

diff --git a/kernel/sched/sched.h b/kernel/sched/sched.h
index d8d3613..1510e08 100644
--- a/kernel/sched/sched.h
+++ b/kernel/sched/sched.h
@@ -208,6 +208,7 @@ struct cfs_rq {
  unsigned long nr_running, h_nr_running;

  u64 exec_clock;
+ u64 prev_exec_clock;
  u64 min_vruntime;
 #ifndef CONFIG_64BIT
  u64 min_vruntime_copy;
@@ -307,6 +308,8 @@ struct rt_rq {
  struct plist_head pushable_tasks;
 #endif
  int rt_throttled;
+ u64 exec_clock;
+ u64 prev_exec_clock;
  u64 rt_time;
  u64 rt_runtime;
  /* Nests inside the rq lock: */
--
1.7.7.4
```