Subject: [PATCH 5/5] expose per-taskgroup schedstats in cgroup
Posted by Glauber Costa on Thu, 02 Feb 2012 14:19:32 GMT
View Forum Message <> Reply to Message

This patch aims at exposing stat information per-cgroup, such as:
 * idle time,
 * iowait time,
 * steal time,
and friends. The ultimate goal is to be able to present a per-container view of
/proc/stat inside a container. With this patch, everything that is needed to do
that is in place, except for number of switches and number of tasks.

I achieve that by hooking into the schedstats framework, so although the
overhead of that is prone to discussion, I am not adding anything, but reusing
what's already there instead. The exception being that the data is now computed
and stored in non-task se's as well, instead of entity_is_task() branches.
However, I expect this to be minimum comparing to the alternative of adding new
hierarchy walks. Those are kept intact.

Note that in this first version, I am using clock_t units, being quite
proc-centric. It made my testing easier, but I am happy to show any units
you guys would prefer.

Signed-off-by: Glauber Costa <glommer@parallels.com>
---
 kernel/sched/core.c | 114 ++
 kernel/sched/fair.c | 45 ++++++++++++++++++++++
 kernel/sched/sched.h | 3 +
 3 files changed, 162 insertions(+), 0 deletions(-)

diff --git a/kernel/sched/core.c b/kernel/sched/core.c
index 013ca9c..fc2b9ed 100644
--- a/kernel/sched/core.c
+++ b/kernel/sched/core.c
@@ -7975,6 +7975,107 @@ static u64 cpu_rt_period_read_uint(struct cgroup *cgrp, struct cftype
*cft)
 }
 #endif /* CONFIG_RT_GROUP_SCHED */

+#ifdef CONFIG_SCHEDSTATS
+
+#ifdef CONFIG_FAIR_GROUP_SCHED
+#define fair_rq(field, tg, i) tg->cfs_rq[i]->field
+#define fair_se(field, tg, i) tg->se[i]->statistics.field
+#else
+#define fair_rq(field, tg, i) 0
+#endif
+

```
+#ifdef CONFIG_RT_GROUP_SCHED
+#define rt_rq(field, tg, i)  tg->rt_rq[i]->field
+#else
+#define rt_rq(field, tg, i)  0
+#endif
+
+static u64 tg_nr_running(struct task_group *tg, int cpu)
+{
+ /*
+  * because of autogrouped groups in root_task_group, the
+  * following does not hold.
+  */
+ if (tg != &root_task_group)
+  return rt_rq(rt_nr_running, tg, cpu) + fair_rq(nr_running, tg, cpu);
+
+ return cpu_rq(cpu)->nr_running;
+}
+
+static u64 tg_idle(struct task_group *tg, int cpu)
+{
+ u64 *cpustat = kcpustat_this_cpu->cpustat;
+ u64 val;
+
+ if (tg != &root_task_group) {
+  val = cfs_read_sleep(tg->se[cpu]);
+  /* If we have rt tasks running, we're not really idle */
+  val -= rt_rq(exec_clock, tg, cpu);
+  val = nsec_to_tick(val);
+ } else
+  val = cpustat[CPUTIME_IDLE];
+
+ return cputime_to_clock_t(val);
+}
+
+static u64 tg_steal(struct task_group *tg, int cpu)
+{
+ u64 *cpustat = kcpustat_this_cpu->cpustat;
+ u64 val = cpustat[CPUTIME_STEAL];
+
+ if (tg != &root_task_group)
+  val = nsec_to_tick(cfs_read_wait(tg->se[cpu]));
+ else
+  val = cpustat[CPUTIME_STEAL];
+
+ return cputime_to_clock_t(val);
+}
+
+static u64 tg_nr_iowait(struct task_group *tg, int cpu)
```

```
+{
+ if (tg != &root_task_group)
+  return atomic_read(&tg->se[cpu]->nr_iowait);
+
+ return atomic_read(&cpu_rq(cpu)->nr_iowait);
+}
+
+static u64 tg_iowait(struct task_group *tg, int cpu)
+{
+ u64 *cpustat = kcpustat_this_cpu->cpustat;
+ u64 val = 0;
+
+ if (tg != &root_task_group)
+  val = nsec_to_tick(cfs_read_iowait(tg->se[cpu]));
+ else
+  val = cpustat[CPUTIME_IOWAIT];
+
+ return cputime_to_clock_t(val);
+}
+
+static int cpu_schedstats_show(struct cgroup *cgrp, struct cftype *cft,
+          struct seq_file *m)
+{
+ struct task_group *tg = cgroup_tg(cgrp);
+ int cpu;
+ /*
+  * TODO: nr_switches is one of the statistics we're interested in, but
+  * it is potentially too heavy on the scheduler.
+  */
+ u64 nr_switches = 0;
+
+ for_each_online_cpu(cpu) {
+  seq_printf(m,
+  "cpu%d %llu %llu %llu %llu %llu\n",
+  cpu, tg_idle(tg, cpu), tg_iowait(tg, cpu), tg_steal(tg, cpu),
+  tg_nr_iowait(tg,cpu), nr_switches,
+  tg_nr_running(tg, cpu)
+  );
+ }
+
+ return 0;
+}
+#endif
+
 static struct cftype cpu_files[] = {
 #ifdef CONFIG_FAIR_GROUP_SCHED
  {
@@ -7982,6 +8083,19 @@ static struct cftype cpu_files[] = {
```

```
   .read_u64 = cpu_shares_read_u64,
   .write_u64 = cpu_shares_write_u64,
  },
+/*
+ * In theory, those could be done using the rt tasks as a basis
+ * as well. Since we're interested in figures like idle, iowait, etc
+ * for the whole cgroup, the results should be the same.
+ * But that only complicates the code, and I doubt anyone using !FAIR_GROUP_SCHED
+ * is terribly interested in those.
+ */
+#ifdef CONFIG_SCHEDSTATS
+ {
+  .name = "schedstat_percpu",
+  .read_seq_string = cpu_schedstats_show,
+ },
+#endif
 #endif
 #ifdef CONFIG_CFS_BANDWIDTH
  {
diff --git a/kernel/sched/fair.c b/kernel/sched/fair.c
index e301ba4..5305bb1 100644
--- a/kernel/sched/fair.c
+++ b/kernel/sched/fair.c
@@ -721,6 +721,41 @@ update_stats_wait_start(struct cfs_rq *cfs_rq, struct sched_entity *se)
  schedstat_set(se->statistics.wait_start, rq_of(cfs_rq)->clock);
 }

+#ifdef CONFIG_SCHEDSTATS
+u64 cfs_read_sleep(struct sched_entity *se)
+{
+ struct cfs_rq *cfs_rq = se->cfs_rq;
+ u64 value = se->statistics.sum_sleep_runtime;
+
+ if (!se->statistics.sleep_start)
+  return value;
+
+ return value + rq_of(cfs_rq)->clock - se->statistics.sleep_start;
+}
+
+u64 cfs_read_iowait(struct sched_entity *se)
+{
+ struct cfs_rq *cfs_rq = se->cfs_rq;
+ u64 value = se->statistics.iowait_sum;
+
+ if (!se->statistics.block_start)
+  return value;
+
+ return value + rq_of(cfs_rq)->clock - se->statistics.block_start;
```

```
+}
+
+u64 cfs_read_wait(struct sched_entity *se)
+{
+ struct cfs_rq *cfs_rq = se->cfs_rq;
+ u64 value = se->statistics.wait_sum;
+
+ if (!se->statistics.wait_start)
+  return value;
+
+ return value + rq_of(cfs_rq)->clock - se->statistics.wait_start;
+}
+#endif
+
 /*
  * Task is being enqueued - update stats:
  */
@@ -1046,6 +1081,10 @@ static void enqueue_sleeper(struct cfs_rq *cfs_rq, struct sched_entity
*se)
    }
    account_scheduler_latency(tsk, delta >> 10, 0);
   }
+  else if (atomic_read(&se->nr_iowait)) {
+   se->statistics.iowait_sum += delta;
+   se->statistics.iowait_count++;
+  }
  }
 #endif
 }
@@ -1199,6 +1238,12 @@ dequeue_entity(struct cfs_rq *cfs_rq, struct sched_entity *se, int
flags)
     se->statistics.sleep_start = rq_of(cfs_rq)->clock;
    if (tsk->state & TASK_UNINTERRUPTIBLE)
     se->statistics.block_start = rq_of(cfs_rq)->clock;
+  } else {
+   if (atomic_read(&se->nr_iowait))
+    se->statistics.block_start = rq_of(cfs_rq)->clock;
+   else
+    se->statistics.sleep_start = rq_of(cfs_rq)->clock;
+
   }
 #endif
  }
diff --git a/kernel/sched/sched.h b/kernel/sched/sched.h
index 53d13dd..7ec2482 100644
--- a/kernel/sched/sched.h
+++ b/kernel/sched/sched.h
@@ -1156,6 +1156,9 @@ extern void init_rt_rq(struct rt_rq *rt_rq, struct rq *rq);
```

```
 extern void unthrottle_offline_cfs_rqs(struct rq *rq);

 extern void account_cfs_bandwidth_used(int enabled, int was_enabled);
+extern u64 cfs_read_sleep(struct sched_entity *se);
+extern u64 cfs_read_iowait(struct sched_entity *se);
+extern u64 cfs_read_wait(struct sched_entity *se);

 #ifdef CONFIG_NO_HZ
 enum rq_nohz_flag_bits {
--
1.7.7.4
```