Subject: Re: [RFC][-mm] Simple stats for cpu resource controller v4
Posted by Li Zefan on Mon, 12 May 2008 02:54:45 GMT
View Forum Message <> Reply to Message

Balaji Rao wrote:
> /*
> @@ -3892,8 +3923,17 @@ void account_system_time(struct task_struct *p, int hardirq_offset,
> cpustat->irq = cputime64_add(cpustat->irq, tmp);
> else if (softirq_count())
> cpustat->softirq = cputime64_add(cpustat->softirq, tmp);
> - else if (p != rq->idle)
> + else if (p != rq->idle) {
> cpustat->system = cputime64_add(cpustat->system, tmp);
> +#ifdef CONFIG_CGROUP_SCHED
> + {
> + struct task_group *tg;
> + tg = task_group(p);
> + __cpu_cgroup_stat_add(tg->stat, CPU_CGROUP_STAT_STIME,
> + cputime_to_msecs(cputime));
> + }
> + }

You should put this '}' after '#endif'

> +#endif
> else if (atomic_read(&rq->nr_iowait) > 0)
> cpustat->iowait = cputime64_add(cpustat->iowait, tmp);
> else
> @@ -8179,10 +8219,26 @@ static inline struct task_group *cgroup_tg(struct cgroup *cgrp)
> struct task_group, css);
> }
>
> +static void cpu_cgroup_initialize(int early)
> +{
> + int i;
> + struct cpu_cgroup_stat *stat;
> +
> + if (!early) {
> + stat = kmalloc(sizeof(struct cpu_cgroup_stat)
> + , GFP_KERNEL);
> + for (i = 0; i < CPU_CGROUP_STAT_NSTATS; i++)
> + percpu_counter_init(
> + &stat->cpustat[i], 0);
> + init_task_group.stat = stat;
> + }
> +}
> +
> static struct cgroup_subsys_state *

> cpu_cgroup_create(struct cgroup_subsys *ss, struct cgroup *cgrp)
> {
> struct task_group *tg;
> + int i;
>
> if (!cgrp->parent) {
> /* This is early initialization for the top cgroup */
> @@ -8198,6 +8254,10 @@ cpu_cgroup_create(struct cgroup_subsys *ss, struct cgroup *cgrp)
> if (IS_ERR(tg))
> return ERR_PTR(-ENOMEM);
>
> + tg->stat = kmalloc(sizeof(struct cpu_cgroup_stat), GFP_KERNEL);
> + for (i = 0; i < CPU_CGROUP_STAT_NSTATS; i++)
> + percpu_counter_init(&tg->stat->cpustat[i], 0);
> +

I guess you forgot to free those things in cpu_cgroup_destroy().

> /* Bind the cgroup to task_group object we just created */
> tg->css.cgroup = cgrp;
>
> @@ -8251,6 +8311,38 @@ static u64 cpu_shares_read_u64(struct cgroup *cgrp, struct cftype
*cft)
> }
> #endif
>
> +static s64 cpu_cgroup_read_stat(struct cpu_cgroup_stat *stat,
> + enum cpu_cgroup_stat_index idx)
> +{
> + if (stat)
> + return percpu_counter_read(&stat->cpustat[idx]);
> +
> + return 0;
> +}
> +
> +static const struct cpu_cgroup_stat_desc {
> + const char *msg;
> + u64 unit;
> +} cpu_cgroup_stat_desc[] = {
> + [CPU_CGROUP_STAT_UTIME] = { "utime", 1, },
> + [CPU_CGROUP_STAT_STIME] = { "stime", 1, },
> +};
> +
> +static int cpu_cgroup_stats_show(struct cgroup *cgrp, struct cftype *cft,
> + struct cgroup_map_cb *cb)
> +{
> + struct task_group *tg = cgroup_tg(cgrp);
> + struct cpu_cgroup_stat *stat = tg->stat;

```
> + int i;
> + for (i = 0; i < CPU_CGROUP_STAT_NSTATS; i++) {
> +  s64 val;
> +  val = cpu_cgroup_read_stat(stat, i);
> +  val *= cpu_cgroup_stat_desc[i].unit;
> +  cb->fill(cb, cpu_cgroup_stat_desc[i].msg, val);
> + }
> + return 0;
> +}
> +
>  #ifdef CONFIG_RT_GROUP_SCHED
>  static ssize_t cpu_rt_runtime_write(struct cgroup *cgrp, struct cftype *cft,
>      s64 val)
> @@ -8295,6 +8387,10 @@ static struct cftype cpu_files[] = {
>    .write_u64 = cpu_rt_period_write_uint,
>   },
>  #endif
> + {
> +  .name = "stat",
> +  .read_map = cpu_cgroup_stats_show,
> + },
>  };
>
>  static int cpu_cgroup_populate(struct cgroup_subsys *ss, struct cgroup *cont)
> @@ -8304,6 +8400,7 @@ static int cpu_cgroup_populate(struct cgroup_subsys *ss, struct
cgroup *cont)
>
>  struct cgroup_subsys cpu_cgroup_subsys = {
>   .name  = "cpu",
> + .initialize = cpu_cgroup_initialize,
>   .create  = cpu_cgroup_create,
>   .destroy = cpu_cgroup_destroy,
>   .can_attach = cpu_cgroup_can_attach,
> --
```