
Subject: [PATCH COMMIT] diff-fairsched-ve-20060530

Posted by [xemul](#) on Tue, 30 May 2006 14:32:43 GMT

[View Forum Message](#) <> [Reply to Message](#)

Added to 026test013

Patch from OpenVZ team <devel@openvz.org>
Virtualization fixes in fairsched.

This includes capability tuning, some per-ve statistics
and /proc/fairsched file with old-format data that may
be needed by some utils (vzcpucheck at least).

http://bugzilla.openvz.org/show_bug.cgi?id=176

```
--- ./include/linux/fairsched.h.vemix 2006-04-21 16:25:26.000000000 +0400
+++ ./include/linux/fairsched.h 2006-04-21 16:25:27.000000000 +0400
@@ -57,6 +58,9 @@ struct fairsched_node {
```

```
    struct list_head nodelist;
    int id;
+#ifdef CONFIG_VE
+ struct ve_struct *owner_env;
+#endif
    struct vcpu_scheduler *vsched;
};
```

```
--- ./kernel/fairsched.c.vemix 2006-04-21 16:25:26.000000000 +0400
+++ ./kernel/fairsched.c 2006-04-21 16:25:27.000000000 +0400
@@ -174,6 +174,9 @@ static fschvalue_t max_value;
```

```
struct fairsched_node fairsched_init_node = {
    .id = INT_MAX,
+#ifdef CONFIG_VE
+ .owner_env = get_ve0(),
+#endif
    .weight = 1,
};
EXPORT_SYMBOL(fairsched_init_node);
@@ -192,6 +195,15 @@ static fschtag_t max_latency;
```

```
static DECLARE_MUTEX(fairsched_mutex);
```

```
+ /*****
+ */
+ * Small helper routines
+ */
```

```

+ /*****
+
+ /* this didn't proved to be very valuable statistics... */
+#define fairsched_inc_ve_strv(node, cycles) do {} while(0)
+#define fairsched_dec_ve_strv(node, cycles) do {} while(0)

/*****
/*
@@ -455,11 +482,13 @@ void fairsched_decrun(struct fairsched_n
void fairsched_inccpu(struct fairsched_node *node)
{
node->nr_pcpu++;
+ fairsched_dec_ve_strv(node, cycles);
}

static inline void __fairsched_deccpu(struct fairsched_node *node)
{
node->nr_pcpu--;
+ fairsched_inc_ve_strv(node, cycles);
}

void fairsched_deccpu(struct fairsched_node *node)
@@ -476,6 +505,9 @@ static void fairsched_account(struct fai
fschdur_t duration;

duration = FSCHDURATION(time, __get_cpu_var(prev_schedule));
+#ifdef CONFIG_VE
+ CYCLES_DADD(&node->owner_env->cpu_used_ve, duration);
+#endif

/*
* The duration is not greater than TICK_DUR since
@@ -576,6 +608,9 @@ static int do_fairsched_mknod(unsigned i
node->weight = weight;
INIT_LIST_HEAD(&node->runlist);
node->id = newid;
+#ifdef CONFIG_VE
+ node->owner_env = get_exec_env();
+#endif

spin_lock_irq(&fairsched_lock);
list_add(&node->nodelist, &fairsched_node_head);
@@ -593,7 +628,7 @@ asmlinkage int sys_fairsched_mknod(unsig
{
int retval;

- if (!capable(CAP_SYS_NICE))
+ if (!capable(CAP_SETVEID))

```

```

return -EPERM;

down(&fairsched_mutex);
@@ -637,7 +672,7 @@ asmlinkage int sys_fairsched_rmnod(unsig
{
int retval;

- if (!capable(CAP_SYS_NICE))
+ if (!capable(CAP_SETVEID))
return -EPERM;

down(&fairsched_mutex);
@@ -673,7 +708,7 @@ asmlinkage int sys_fairsched_chwt(unsign
{
int retval;

- if (!capable(CAP_SYS_NICE))
+ if (!capable(CAP_SETVEID))
return -EPERM;

down(&fairsched_mutex);
@@ -719,9 +754,12 @@ int do_fairsched_rate(unsigned int id, i
retval = node->rate;
break;
case 1:
- node->rate = 0; /* XXX not needed, but was added by
- a special patch. I'm too lazy
- to try to find out why --SAW */
+ node->rate = 0; /* This assignment is not needed
+ for the kernel code, and it should
+ not rely on rate being 0 when it's
+ unset. This is a band-aid for some
+ existing tools (don't know which one
+ exactly). --SAW */
node->rate_limited = 0;
node->value = max_value;
if (node->delayed) {
@@ -749,7 +787,7 @@ asmlinkage int sys_fairsched_rate(unsign
{
int retval;

- if (!capable(CAP_SYS_NICE))
+ if (!capable(CAP_SETVEID))
return -EPERM;

down(&fairsched_mutex);
@@ -810,7 +848,7 @@ asmlinkage int sys_fairsched_mvpr(pid_t
{

```

```

int retval;

- if (!capable(CAP_SYS_NICE))
+ if (!capable(CAP_SETVEID))
    return -EPERM;

    down(&fairsched_mutex);
@@ -829,6 +867,9 @@ EXPORT_SYMBOL(sys_fairsched_mvpr);
/*****

struct fairsched_node_dump {
+#ifdef CONFIG_VE
+ envid_t veid;
+#endif
    int id;
    unsigned weight;
    unsigned rate;
@@ -840,36 +881,53 @@ struct fairsched_node_dump {
    int nr_ready;
    int nr_runnable;
    int nr_pcpu;
+ int nr_tasks, nr_runtasks;
};

struct fairsched_dump {
- int len;
+ int len, compat;
    struct fairsched_node_dump nodes[0];
};

-static struct fairsched_dump *fairsched_do_dump(void)
+static struct fairsched_dump *fairsched_do_dump(int compat)
{
    int nr_nodes;
- int len;
+ int len, i;
    struct fairsched_dump *dump;
    struct fairsched_node *node;
    struct fairsched_node_dump *p;
    unsigned long flags;

start:
- nr_nodes = fairsched_nr_nodes + 16;
+ nr_nodes = (ve_is_super(get_exec_env()) ? fairsched_nr_nodes + 16 : 1);
    len = sizeof(*dump) + nr_nodes * sizeof(dump->nodes[0]);
    dump = ub_vmalloc(len);
    if (dump == NULL)
        goto out;

```

```

spin_lock_irqsave(&fairsched_lock, flags);
- if (nr_nodes < fairsched_nr_nodes)
+ if (ve_is_super(get_exec_env()) && nr_nodes < fairsched_nr_nodes)
    goto repeat;
p = dump->nodes;
list_for_each_entry_reverse(node, &fairsched_node_head, nodelist) {
    if ((char *)p - (char *)dump >= len)
        break;
+ p->nr_tasks = 0;
+ p->nr_runtasks = 0;
+#ifdef CONFIG_VE
+ if (!ve_accessible(node->owner_env, get_exec_env()))
+     continue;
+ p->veid = node->owner_env->veid;
+ if (compat) {
+     p->nr_tasks = atomic_read(&node->owner_env->pcounter);
+     for (i = 0; i < NR_CPUS; i++)
+         p->nr_runtasks +=
+             VE_CPU_STATS(node->owner_env, i)
+                 ->nr_running;
+     if (p->nr_runtasks < 0)
+         p->nr_runtasks = 0;
+ }
+#endif
    p->id = node->id;
    p->weight = node->weight;
    p->rate = node->rate;
@@ -884,6 +942,7 @@ start:
    p++;
}
dump->len = p - dump->nodes;
+ dump->compat = compat;
spin_unlock_irqrestore(&fairsched_lock, flags);

```

out:

```
@@ -897,6 +956,102 @@ repeat:
```

```
#define FAIRSCHED_PROC_HEADLINES 2
```

```

+#if defined(CONFIG_VE)
+/*
+ * File format is dictated by compatibility reasons.
+ */
+static int fairsched_seq_show(struct seq_file *m, void *v)
+{
+ struct fairsched_dump *dump;
+ struct fairsched_node_dump *p;

```

```

+ unsigned vid, nid, pid, r;
+
+ dump = m->private;
+ p = (struct fairsched_node_dump *)((unsigned long)v & ~3UL);
+ if (p - dump->nodes < FAIRSCHED_PROC_HEADLINES) {
+ if (p == dump->nodes)
+ seq_printf(m, "Version: 2.6 debug\n");
+ else if (p == dump->nodes + 1)
+ seq_printf(m,
+ "    veid "
+ "    id "
+ "    parent "
+ "weight "
+ " rate "
+ "tasks "
+ " run "
+ "cpus"
+ " "
+ "flg "
+ "ready "
+ "    start_tag "
+ "    value "
+ "    delay"
+ "\n");
+ } else {
+ p -= FAIRSCHED_PROC_HEADLINES;
+ vid = nid = pid = 0;
+ r = (unsigned long)v & 3;
+ if (p == dump->nodes) {
+ if (r == 2)
+ nid = p->id;
+ } else {
+ if (!r)
+ nid = p->id;
+ else if (r == 1)
+ vid = pid = p->id;
+ else
+ vid = p->id, nid = 1;
+ }
+ seq_printf(m,
+ "%10u "
+ "%10u %10u %6u %5u %5u %5u %4u"
+ " "
+ " %c%c %5u %20Lu %20Lu %20Lu"
+ "\n",
+ vid,
+ nid,
+ pid,

```

```

+     p->weight,
+     p->rate,
+     p->nr_tasks,
+     p->nr_runtasks,
+     p->nr_pcpu,
+     p->rate_limited ? 'L' : '',
+     p->delayed ? 'D' : '',
+     p->nr_ready,
+     p->start_tag.t,
+     p->value.v,
+     p->delay
+ );
+ }
+
+ return 0;
+}
+
+static void *fairsched_seq_start(struct seq_file *m, loff_t *pos)
+{
+ struct fairsched_dump *dump;
+ unsigned long l;
+
+ dump = m->private;
+ if (*pos >= dump->len * 3 - 1 + FAIRSCHED_PROC_HEADLINES)
+ return NULL;
+ if (*pos < FAIRSCHED_PROC_HEADLINES)
+ return dump->nodes + *pos;
+ /* guess why... */
+ l = (unsigned long)(dump->nodes +
+ ((unsigned long)*pos + FAIRSCHED_PROC_HEADLINES * 2 + 1) / 3);
+ l |= ((unsigned long)*pos + FAIRSCHED_PROC_HEADLINES * 2 + 1) % 3;
+ return (void *)l;
+}
+static void *fairsched_seq_next(struct seq_file *m, void *v, loff_t *pos)
+{
+ ++*pos;
+ return fairsched_seq_start(m, pos);
+}
+#endif
+
+static int fairsched2_seq_show(struct seq_file *m, void *v)
+{
+ struct fairsched_dump *dump;
@@ -970,6 +1125,14 @@ static void fairsched2_seq_stop(struct s
+{
+}
+}

+#ifdef CONFIG_VE

```

```

+static struct seq_operations fairsched_seq_op = {
+ .start = fairsched_seq_start,
+ .next = fairsched_seq_next,
+ .stop = fairsched2_seq_stop,
+ .show = fairsched_seq_show
+};
+#endif
static struct seq_operations fairsched2_seq_op = {
 .start = fairsched2_seq_start,
 .next = fairsched2_seq_next,
@@ -980,12 +1143,19 @@ static int fairsched_seq_open(struct inode
{
 int ret;
 struct seq_file *m;
+ int compat;

- ret = seq_open(file, &fairsched2_seq_op);
+#ifdef CONFIG_VE
+ compat = (file->f_dentry->d_name.len == sizeof("fairsched") - 1);
+ ret = seq_open(file, compat ? &fairsched_seq_op : &fairsched2_seq_op);
+#else
+ compat = 0;
+ ret = seq_open(file, fairsched2_seq_op);
+#endif
 if (ret)
 return ret;
 m = file->private_data;
- m->private = fairsched_do_dump();
+ m->private = fairsched_do_dump(compat);
 if (m->private == NULL) {
 seq_release(inode, file);
 ret = -ENOMEM;
@@ -1065,7 +1235,10 @@ void __init fairsched_init_late(void)
 fairsched_calibrate();
 fairsched_recompute_max_latency();

- entry = create_proc_entry("fairsched2", S_IRUGO, NULL);
+ entry = create_proc_glob_entry("fairsched", S_IRUGO, NULL);
+ if (entry)
+ entry->proc_fops = &proc_fairsched_operations;
+ entry = create_proc_glob_entry("fairsched2", S_IRUGO, NULL);
 if (entry)
 entry->proc_fops = &proc_fairsched_operations;
}

```

File Attachments

1) [diff-fairsched-ve-20060530](#), downloaded 456 times