
Subject: [PATCH] vzctl patch to set range of veid
Posted by [leiqunni](#) on Tue, 03 May 2011 01:37:10 GMT
[View Forum Message](#) <> [Reply to Message](#)

```
include/vzctl.h | 84 ++++++  
src/vzctl.c    | 323 ++++++-----  
2 files changed, 330 insertions(+), 77 deletions(-)
```

```
diff --git a/include/vzctl.h b/include/vzctl.h  
index 4357532..65ae305 100644  
--- a/include/vzctl.h  
+++ b/include/vzctl.h  
@@ -15,6 +15,9 @@  
 * along with this program; if not, write to the Free Software  
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA  
 */  
+  
+/* patch to set range of veid by leiqunni May 3, 2011 */  
+#ifndef _VZCTL_H_  
#define _VZCTL_H_  
  
@@ -53,4 +56,85 @@ enum {  
    SET_RESTART,  
};  
  
+/* from vzlist.h */  
+#define HAVE_VZLIST_IOCTL 0  
+  
+#define PROCVESTAT    "/proc/vz/vestat"  
+#define PROCUBC       "/proc/user_beancounters"  
+#define PROCQUOTA     "/proc/vz/vzquota"  
+#define PROCFSHED     "/proc/fairsched"  
+  
+#define VZQUOTA        "/usr/sbin/vzquota"  
+  
+#define MAXCPUUNITS   500000  
+  
+enum {  
+    VE_RUNNING,  
+    VE_STOPPED,  
+    VE_MOUNTED,  
+    VE_SUSPENDED  
+};  
+  
+struct Cubc {  
+    unsigned long kmemsize[5];
```

```

+     unsigned long lockedpages[5];
+     unsigned long privvmpages[5];
+     unsigned long shmpages[5];
+     unsigned long numproc[5];
+     unsigned long physpages[5];
+     unsigned long vmguarpages[5];
+     unsigned long oomguarpages[5];
+     unsigned long numtcpsock[5];
+     unsigned long numflock[5];
+     unsigned long numpyt[5];
+     unsigned long numsiginfo[5];
+     unsigned long tcpsndbuf[5];
+     unsigned long tcprcvbuf[5];
+     unsigned long othersockbuf[5];
+     unsigned long dgramrcvbuf[5];
+     unsigned long numothersock[5];
+     unsigned long dcachesize[5];
+     unsigned long numfile[5];
+     unsigned long numiptent[5];
+     unsigned long swappages[5];
+};

+
+struct Cquota {
+    unsigned long diskspace[3]; // 0-usage 1-softlimit 2-hardlimit
+    unsigned long diskinodes[3]; // 0-usage 1-softlimit 2-hardlimit
+};

+
+struct Ccpustat {
+    float la[3]; // load average
+    float uptime;
+};

+
+struct Ccpu {
+    unsigned long limit[2]; // 0-limit, 1-units
+};

+
+struct Cio {
+    int ioprio;
+};

+
+struct Cveinfo {
+    int veid;
+    char *hostname;
+    char *name;
+    char *description;
+    char *ostemplate;
+    char *ip;
+    char *ve_private;

```

```

+     char *ve_root;
+     struct Cubc *ubc;
+     struct Cquota *quota;
+     struct Ccpustat *cpustat;
+     struct Ccpu *cpu;
+     struct Cio io;
+     int status;
+     int hide;
+     int onboot;
+     unsigned long *bootorder;
+};
+
#endif
diff --git a/src/vzctl.c b/src/vzctl.c
index 273a490..24771f4 100644
--- a/src/vzctl.c
+++ b/src/vzctl.c
@@ -16,12 +16,15 @@
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
 */
/* patch to set range of veid by leiqunni May 3, 2011 */
+
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <getopt.h>
#include <signal.h>
#include <limits.h>
+#include <dirent.h>

#include "vzctl.h"
#include "env.h"
@@ -33,6 +36,25 @@
#include "util.h"
#include "modules.h"

+struct Cveinfo *veinfo = NULL;
+int n_veinfo = 0;
+int *g_ve_list = NULL;
+int n_ve_list = 0;
+int veid_range[256] = {0};
+int n_range = 0;
+
+int get_ve_list();
+int id_sort_fn(const void *val1, const void *val2);
+int check_veid_restr(int veid);
+void add_elem(struct Cveinfo *ve);

```

```

+int veid_search_fn(const void* val1, const void* val2);
+void *x_realloc(void *ptr, int size);
+
+void veid_parse_range(char *str);
+void veid_parse_range_comma(char *str);
+void veid_parse_range_one(char *str);
+void veid_parse_range_dash(char *str);
+
struct mod_action g_action;
char *_proc_title;
int _proc_title_len;
@@ -101,6 +123,18 @@ void usage(int rc)
" --numpy N[:N] --numsiginfo N[:N] --dcachesize N[:N]\n"
" --numiptent N[:N] --physpages P[:P] --avnumproc N[:N]\n"
" --swappages P[:P]\n"
+"\\n"
+"ctid is made up of one range, or many ranges separated by commas.\n"
+"Selected input is written in the same order that it is read,\n"
+"And is written exactly once. Each range is one of:\n"
+"\\n"
+" all All ctid\n"
+" N N\n"
+" N- from N to end of ctid\n"
+" N-M from N to M (included) ctid\n"
+" -M from first to M ctid\n"
+" N,M N and M\n"
+" I,J,N-M I and J and from N to M\n"
);
    memset(&mod, 0, sizeof(mod));
    set_log_level(0);
@@ -119,7 +153,9 @@ int main(int argc, char *argv[], char *envp[])
    int quiet = 0;
    int veid, ret, skiplock = 0;
    char buf[256];
-    vps_param *gparam = NULL, *vps_p = NULL, *cmd_p = NULL;
+
+    vps_param *cmd_p = init_vps_param();
+
    const char *action_nm;
    struct sigaction act;
    char *name = NULL, *opt;
@@ -127,10 +163,6 @@ int main(int argc, char *argv[], char *envp[])
    _proc_title = argv[0];
    _proc_title_len = envp[0] - argv[0];
-
-    gparam = init_vps_param();
-    vps_p = init_vps_param();
-    cmd_p = init_vps_param();

```

```

    sigemptyset(&act.sa_mask);
    act.sa_handler = SIG_IGN;
    act.sa_flags = 0;
@@ -141,7 +173,7 @@ int main(int argc, char *argv[], char *envp[])
    if (!strcmp(opt, "--verbose")) {
        if (argc > 2 &&
            !parse_int(argv[2], &verbose_tmp))
-           !parse_int(argv[2], &verbose_tmp))
+           {
+               verbose += verbose_tmp;
+               argc--; argv++;
@@ -223,95 +255,232 @@ int main(int argc, char *argv[], char *envp[])
    if (!g_action.mod_count) {
        fprintf(stderr, "Bad command: %s\n", argv[1]);
        ret = VZ_INVALID_PARAMETER_SYNTAX;
-       goto error;
+       goto error_;
    }
}
if (argc < 3) {
    fprintf(stderr, "CT ID missing\n");
    ret = VZ_INVALID_PARAMETER_VALUE;
-   goto error;
}
if (parse_int(argv[2], &veid)) {
-   name = strdup(argv[2]);
-   veid = get_veid_by_name(name);
-   if (veid < 0 || veid == INT_MAX) {
-       fprintf(stderr, "Bad CT ID %s\n", argv[2]);
-       ret = VZ_INVALID_PARAMETER_VALUE;
-       goto error;
-   }
+   goto error_;
}
+
+   veid_parse_range(argv[2]);
+
argc -= 2; argv += 2;
/* Read global config file */
if (vps_parse_config(veid, GLOBAL_CFG, gparam, &g_action)) {
    fprintf(stderr, "Global configuration file %s not found\n",
            GLOBAL_CFG);
    ret = VZ_NOCONFIG;
-   goto error;
}
init_log(gparam->log.log_file, veid, gparam->log.enable != NO,

```

```

-     gparam->log.level, quiet, "vzctl");
- /* Set verbose level from global config if not overwritten
- by --verbose
- */
- if (!verbose_custom && gparam->log.verbose != NULL) {
-     verbose = *gparam->log.verbose;
-     verbose_custom = 1;
- }
- if (verbose < -1)
-     verbose = -1;
- if (verbose_custom)
-     set_log_verbose(verbose);
- if ((ret = parse_action_opt(veid, action, argc, argv, cmd_p,
-     action_nm)))
{
-     goto error;
}
if (veid == 0 && action != ACTION_SET) {
    fprintf(stderr, "Only set actions are allowed for CT0\n");
    ret = VZ_INVALID_PARAMETER_VALUE;
    goto error;
} else if (veid < 0) {
    fprintf(stderr, "Bad CT ID %d\n", veid);
    ret = VZ_INVALID_PARAMETER_VALUE;
    goto error;
}
get_vps_conf_path(veid, buf, sizeof(buf));
if (stat_file(buf)) {
    if (vps_parse_config(veid, buf, vps_p, &g_action)) {
        logger(-1, 0, "Error in config file %s",
               buf);
+    int i;
+    for (i = 0; i < n_range; i++) {
+        veid = veid_range[i];
+        fprintf(stderr, "CTID:%d ", veid);
+
+        vps_param *gparam = init_vps_param();
+        vps_param *vps_p = init_vps_param();
+//        vps_param *cmd_p = init_vps_param();
+
+        /* Read global config file */
+        if (vps_parse_config(veid, GLOBAL_CFG, gparam, &g_action)) {
+            fprintf(stderr, "Global configuration file %s
not found\n", GLOBAL_CFG);
+            ret = VZ_NOCONFIG;
+            goto error;
}
-        if (name != NULL &&

```

```

-     vps_p->res.name.name != NULL &&
-         strcmp(name, vps_p->res.name.name))
+     init_log(gparam->log.log_file, veid, gparam->log.enable != NO,
+             gparam->log.level, quiet, "vzctl");
+     /* Set verbose level from global config if not overwritten
+        by --verbose
+     */
+     if (!verbose_custom && gparam->log.verbose != NULL) {
+         verbose = *gparam->log.verbose;
+         verbose_custom = 1;
+     }
+     if (verbose < -1)
+         verbose = -1;
+     if (verbose_custom)
+         set_log_verbose(verbose);
+     if ((ret = parse_action_opt(veid, action, argc, argv,
cmd_p, action_nm)))
{
-         logger(-1, 0, "Unable to find container by name %s",
-                 name);
-         ret = VZ_INVALID_PARAMETER_VALUE;
-         goto error;
}
- } else if (action != ACTION_CREATE &&
-             action != ACTION_STATUS &&
-             action != ACTION_SET)
{
-     logger(-1, 0, "Container config file does not exist");
-     ret = VZ_NOVECONFIG;
-     goto error;
}
- merge_vps_param(gparam, vps_p);
- merge_global_param(cmd_p, gparam);
- ret = run_action(veid, action, gparam, vps_p, cmd_p, argc-1, argv+1,
-                 skiplock);
+     if (veid == 0 && action != ACTION_SET) {
+         fprintf(stderr, "Only set actions are allowed
for CT0\n");
+         ret = VZ_INVALID_PARAMETER_VALUE;
+     } else if (veid < 0) {
+         fprintf(stderr, "Bad CT ID %d\n", veid);
+         ret = VZ_INVALID_PARAMETER_VALUE;
+     }
+     get_vps_conf_path(veid, buf, sizeof(buf));
+     if (stat_file(buf)) {
+         if (vps_parse_config(veid, buf, vps_p, &g_action)) {
+             logger(-1, 0, "Error in config file %s", buf);
+             ret = VZ_NOCONFIG;

```

```

+
+        }
+        if (name != NULL &&
+            vps_p->res.name.name != NULL &&
+            strcmp(name, vps_p->res.name.name))
+        {
+            logger(-1, 0, "Unable to find
container by name %s", name);
+            ret = VZ_INVALID_PARAMETER_VALUE;
+        }
+    } else if (action != ACTION_CREATE &&
+               action != ACTION_STATUS &&
+               action != ACTION_SET)
+    {
+        logger(-1, 0, "Container config file does not exist");
+        ret = VZ_NOVECONFIG;
+    }
+    merge_vps_param(gparam, vps_p);
+    merge_global_param(cmd_p, gparam);
+    ret = run_action(veid, action, gparam, vps_p, cmd_p,
argc-1, argv+1, skiplock);

error:
+
+        free_vps_param(gparam);
+        free_vps_param(vps_p);
+//        free_vps_param(cmd_p);
+
+}
+
+error_:
+
+        free_modules(&g_action);
-        free_vps_param(gparam);
-        free_vps_param(vps_p);
-        free_vps_param(cmd_p);
-        free_log();
-        free(name);

return ret;
}

+
+int get_ve_list()
+{
+    DIR *dp;
+    struct dirent *ep;
+    int veid, res;
+    struct Cveinfo ve;
+    char str[6];
+
+    dp = opendir(VPS_CONF_DIR);
+    if (dp == NULL) {

```

```

+         return -1;
+
+     }
+     memset(&ve, 0, sizeof(struct Cveinfo));
+     ve.status = VE_STOPPED;
+     while ((ep = readdir (dp))) {
+         res = sscanf(ep->d_name, "%d.%5s", &veid, str);
+         if (!(res == 2 && !strcmp(str, "conf")))
+             continue;
+         if (!check_veid_restr(veid))
+             continue;
+         ve.veid = veid;
+         add_elem(&ve);
+
+     }
+     closedir(dp);
+     if (veinfo != NULL)
+         qsort(veinfo, n_veinfo, sizeof(struct Cveinfo), id_sort_fn);
+     return 0;
+}
+
+int id_sort_fn(const void *val1, const void *val2)
+{
+    int ret;
+    ret = (((const struct Cveinfo*)val1)->veid >
+           ((const struct Cveinfo*)val2)->veid);
+    return ret;
+}
+
+int check_veid_restr(int veid)
+{
+    if (g_ve_list == NULL)
+        return 1;
+    return (bsearch(&veid, g_ve_list, n_ve_list,
+                   sizeof(*g_ve_list), veid_search_fn) != NULL);
+}
+
+void add_elem(struct Cveinfo *ve)
+{
+    veinfo = (struct Cveinfo *)x_realloc(veinfo,
+                                         sizeof(struct Cveinfo) * ++n_veinfo);
+    memcpy(&veinfo[n_veinfo - 1], ve, sizeof(struct Cveinfo));
+    return;
+}
+
+int veid_search_fn(const void* val1, const void* val2)
+{
+    return (*const int *)val1 - *(const int *)val2;
+}
+

```

```

+void *x_realloc(void *ptr, int size)
+{
+    void *tmp;
+
+    if ((tmp = realloc(ptr, size)) == NULL) {
+        printf("Error: unable to allocate %d bytes\n", size);
+        exit(1);
+    }
+    return tmp;
+}
+
+void veid_parse_range(char *str)
+{
+    get_ve_list();
+
+    if (strchr(str, ',')) {
+        veid_parse_range_comma(str);
+    } else {
+        veid_parse_range_one(str);
+    }
+}
+
+void veid_parse_range_comma(char *str)
+{
+    char delims[] = ",";
+    char *veid_str = NULL;
+
+    veid_str = strtok(str, delims);
+
+    if (veid_str == NULL) { // str has no comma.
+        veid_parse_range_one(str);
+    } else {
+        while (veid_str != NULL) {
+            veid_parse_range_one(veid_str);
+            veid_str = strtok(NULL, delims);
+        }
+    }
+}
+
+void veid_parse_range_one(char *str)
+{
+    int veid;
+
+    if (strcmp(str, "all") == 0 || strchr(str, '-') != NULL) { // str is
+'all' or has '-'?
+        veid_parse_range_dash(str);
+    } else {
+        if (parse_int(str, &veid)) {

```

```

+
+     veid = get_veid_by_name(str);
+     if (veid < 0 || veid == INT_MAX) {
+         fprintf(stderr, "Bad CT ID %s\n",str);
+         ret = VZ_INVALID_PARAMETER_VALUE;
+         goto error;
+     }
+
+     veid_range[n_range++] = veid;
+
+}
+
+void veid_parse_range_dash(char *str)
+{
+    int s = -1, e = -1;
+    int n = sscanf(str, "%d-%d", &s, &e);
+
+    if (strcmp(str, "all") == 0) {           // all
+        s = 0; e = INT_MAX;
+    } else if (n == 2 && s > 0 && e > 0) { // ex. 1234-5678
+        s = s; e = e;
+    } else if (n == 1 && s > 0 && e < 0) { // ex. 1234-
+        s = s;
+        e = INT_MAX;
+    } else if (n == 1 && s < 0 && e < 0) { // ex. -5678
+        e = -1 * s; s = 0;
+    } else if (n == 0 && s < 0 && e < 0) { // ex. invalid
+        s = 0; e = 0;
+    }
+
+    for (n = 0; n < n_veinfo; n++) {
+        if (veinfo[n].veid != 0 && veinfo[n].veid >= s &&
+            veinfo[n].veid <= e) {
+            veid_range[n_range++] = veinfo[n].veid;
+        }
+    }
+}
--
```

1.7.4.1
