
Subject: Re: SNMP monitoring of OpenVZ clusters
Posted by [Poikilotherm](#) on Thu, 01 Jul 2010 12:01:09 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hey, I am revamping this thread as I have this problem, too.
I thought about a solution and invented a patch for the proc-module of the ucd-snmp-module of net-snmp.

It basically checks if the compared proc has the same envID as the snmpd process (which is 0 for HN, and VEID for VE).
Via this it is possible to differ between processes of the HN and VEs. In a VE it is does not make sense to use this, because you will not see any other VEs processes. But anyway, it does no harm inside a VE...

I would appreciate some comments and if this also works for you

```
--- a/agent/mibgroup/ucd-snmp/proc.c 2008-06-05 23:11:53.000000000 +0200
+++ b/agent/mibgroup/ucd-snmp/proc.c 2010-07-01 02:33:29.066790818 +0200
@@ -439,14 +439,50 @@
    DIR *dir;
    char cmdline[512], *tmpc;
    char state[64];
+   char envIDStr[32], *tmpid;
    struct dirent *ent;
#endif USE_PROC_CMDLINE
    int fd;
#endif
    int len,plen=strlen(procname),total = 0;
    FILE *status;
+
+   /* var to save the envID */
+   int envID, actEnvID;

+   DEBUGMSGTL(("proc", "opendir /proc.\n"));
    if ((dir = opendir("/proc")) == NULL) return -1;
+
+   /* get our envID out of the /proc/self/status */
+   /* read /proc/self/status */
+   if ((status = fopen("/proc/self/status", "r")) == NULL) {
+       DEBUGMSGTL(("proc", "Could not open /proc/self/status.\n"));
+       return -1;
+   }
+   /* read lines till we find the envID line */
+   envIDStr[0] = '\0';
+   while( strstr(envIDStr, "envID:") == NULL ) {
+       /*if unreadable or EOF, return an error (-1)*/
+       if (fgets(envIDStr, sizeof(envIDStr), status) == NULL) {
```

```

+ fclose(status);
+ DEBUGMSGTL(("proc", "Could not skip lines until \"envID\" found - unreadable or no such line
in status-file.\n"));
+ return -1;
+ }
+ }
+ fclose(status);
+ /* parse this line, get envID */
+ envIDstr[sizeof(envIDstr)-1] = '\0';
+ tmpid = skip_token(envIDstr);
+ if (!tmpid)
+ return -1;
+ for (len=0;; len++) {
+ if (tmpid[len] && isgraph(tmpid[len])) continue;
+ tmpid[len]='\0';
+ break;
+ }
+ envID = atoi(tmpid);
+ DEBUGMSGTL(("proc", "SNMPD is using envID %d.\n", envID));
+
while (NULL != (ent = readdir(dir))) {
    if(!(ent->d_name[0] >= '0' && ent->d_name[0] <= '9')) continue;
#ifndef USE_PROC_CMDLINE /* old method */
@@ -463,10 +499,14 @@
#else
/* read /proc/XX/status */
sprintf(cmdline,"/proc/%s/status",ent->d_name);
- if ((status = fopen(cmdline, "r")) == NULL)
+ DEBUGMSGTL(("Trying to match %s", cmdline));
+ if ((status = fopen(cmdline, "r")) == NULL) {
+     DEBUGMSGTL(("proc", "Could not open /proc/%s/status for reading - skipping.\n",
cmdline));
        continue;
+ }
    if (fgets(cmdline, sizeof(cmdline), status) == NULL) {
        fclose(status);
+ DEBUGMSGTL(("proc", "Could not get process name from /proc/%s/status - skipping.\n",
cmdline));
        break;
    }
    /* Grab the state of the process as well
@@ -476,9 +516,23 @@
    if (fgets(state, sizeof(state), status) == NULL) {
        state[0]='\0';
    }
+
+ /* read lines till we find the envID line */
+ envIDstr[0]='\0';

```

```

+     while( strstr(envIDstr, "envID:") == NULL ) {
+         /*if unreadable or EOF, return an error (-1)*/
+         if (fgets(envIDstr, sizeof(envIDstr), status) == NULL) {
+             fclose(status);
+             DEBUGMSGTL(("proc", "Could not skip lines until \"envID\" found - unreadable or no such
line in status-file - skipping.\n"));
+             break;
+         }
+     }
fclose(status);
+
 cmdline[sizeof(cmdline)-1] = '\0';
state[sizeof(state)-1] = '\0';
+     envIDstr[sizeof(envIDstr)-1] = '\0';
+
/* XXX: assumes Name: is first */
if (strncmp("Name:", cmdline, 5) != 0)
    break;
@@ -489,14 +543,33 @@
if (tmpc[len] && isgraph(tmpc[len])) continue;
tmpc[len]='\0';
break;
-
}
+
DEBUGMSGTL(("proc","Comparing wanted %s against %s\n",
procname, tmpc));
-
if(len==plen && !strncmp(tmpc,procname,plen)) {
+
+/* get envID */
+if (strncmp("envID:",envIDstr, 6) != 0)
+break;
+tmpid = skip_token(envIDstr);
+if (!tmpid)
+break;
+for (len=0; len++) {
+if (tmpid[len] && isgraph(tmpid[len])) continue;
+tmpid[len]='\0';
+break;
+
actEnvID = atoi(tmpid);
+DEBUGMSGTL(("proc", "Compared process has envID: %d.\n", actEnvID));
+
+
if(strlen(tmpc)==plen && !strncmp(tmpc,procname,plen)) {
/* Do not count zombie process as they are not running processes */
if ( strstr(state, "zombie") == NULL ) {
-
    total++;
-
    DEBUGMSGTL(("proc", " Matched. total count now=%d\n", total));
+
if(actEnvID == envID) {

```

```
+     total++;
+     DEBUGMSGTL(("proc", " Matched. total count now=%d\n", total));
+ } else {
+     DEBUGMSGTL(("proc", " Skipping process as it belongs to another envID (=is in another
VE or non HE).\n"));
+ }
} else {
    DEBUGMSGTL(("proc", " Skipping zombie process.\n"));
}
```
