Subject: SNMP monitoring of OpenVZ clusters
Posted by rapallo on Fri, 27 Feb 2009 13:46:41 GMT
View Forum Message <> Reply to Message

Hi,

we want to build a high availability cluster based on Redhat Cluster Suite and OpenVZ. The host systems are monitored by Zenoss using SNMP. The problem is that SNMP "sees" all processes on a node, host AND guest.
When there is for example a VE with an Apache currently running on host A, Zenoss learns that there is an Apache process on host A. If we switch the VE to host system B the monitoring complains because the apache process on host A seems gone...

Is there any possibility to change the visibility of guest processes in the host system (via a /proc/ switch or a kernel config option for example)? Any other ideas?

Thanks in advance :-)

Subject: Re: SNMP monitoring of OpenVZ clusters
Posted by tomp on Mon, 02 Mar 2009 19:22:52 GMT
View Forum Message <> Reply to Message

Can you not disable the process checks for the hardware nodes, and then run snmpd inside each container to be monitored separately?

Subject: Re: SNMP monitoring of OpenVZ clusters
Posted by rapallo on Tue, 03 Mar 2009 15:38:39 GMT
View Forum Message <> Reply to Message

Hmmm there are processes on the host that we would like to monitor ... like the cluster software itself...

Subject: Re: SNMP monitoring of OpenVZ clusters
Posted by kiddbios on Wed, 08 Apr 2009 19:52:33 GMT
View Forum Message <> Reply to Message

We are having the same issue. Did you find anything on this subject?

Subject: Re: SNMP monitoring of OpenVZ clusters
Posted by rapallo on Wed, 15 Apr 2009 08:39:48 GMT
View Forum Message <> Reply to Message

Unfortunately not yet

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;
 #ifdef 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;
 #ifdef 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"));
         }
```