Budapesti Műszaki és Gazdaságtudományi Egyetem

Virtualizációs technológiák és alkalmazásaik (VIMIAV89)

Virtuális gép működésének vizsgálata

Rendszerhívás követési megoldások használata, instrumentálás

Balogh Mihály Zoltán

1 Bevezetés

Virtuális gépek monitorozására több módszer is rendelkezésre áll : figyelemmel kísérhetjük a virtuális gép állapotát, erőforásfogyasztását magával a Hypervisorral, a futtató környezettel. Erre találhatunk példát a VMWare Server, a VMWare ESX, Microsoft Hyper-V … termékekben.

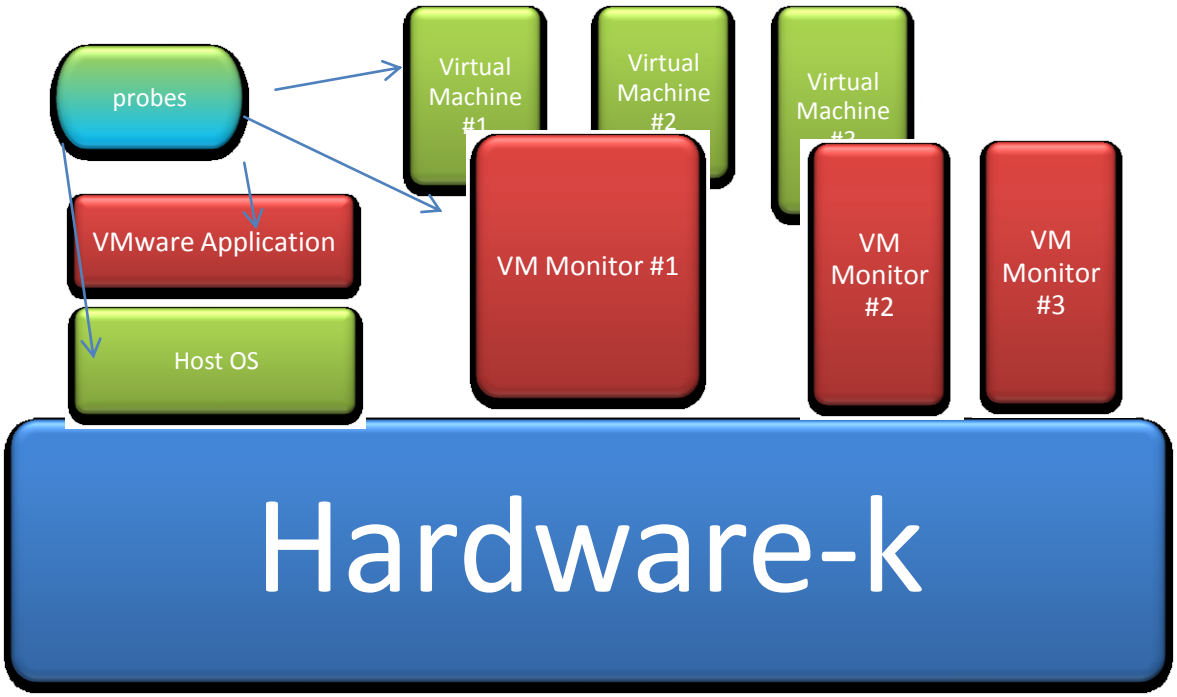
Ha ennél mélyebben szeretnénk belelátni egy virtuális gép működésébe, ahhoz már más eszközökre van szükségünk. Ilyen eszköz lehet egy integrált fejlesztőkörnyezet debuggerével csatlakozni a futó virtuális géphez. Ilyenre van lehetőség az Eclipse és Visual Studio IDE-ek esetében.

Debuggolás esetében a vizsgált program – legyen az virtuális gép vagy bármilyen már program – futása folyamatosan követhető, megállítható. Ennek az ára a megnövekedett erőforrás használat a debugger miatt, illetve az így vizsgált program futása eltérhet a normálistól aminek következtében olyan állapotok maradhatnak rejtve amik csak debugger nélkül jönnének elő.

Eddig csak magáról a virtuális gépről esett szó, azonban egy ilyet általában nem öncélúan csak önmagában használunk. Valamilyen alkalmazást, programkörnyezetet telepítünk egy virtuális gépre. Például adatbázis kiszolgáló, webszerver, különféle operációs rendszerek. Kíváncsiak lehetünk ezen alkalmazások viselkedésére a virtualizált környezeten belül. Erre a már korábban is említett lehetőségek adottak. Most képzeljünk el egy olyan helyzetet, amikor szeretnénk a virtuális gépet és mondjuk a rajta futó Windows operációs rendszert is vizsgálni. Ha mindegyikhez debugger megoldást használunk az igen költséges lehet, ami már komoly többletterhelést jelent a rendszer számára megváltozott viselkedést eredményezve. Vagyis nem a kívánt program futást fogjuk vizsgálni, hanem egy attól eltérőt. Hogy ezt a problémát megoldjuk, olyan megoldásra van szükségünk amivel a virtualizált környezetet, a virtualizáló környezetet is tudjuk figyelni viszont ez nem okoz számottevő plusz terhelést a rendszernek. Egy ilyen megoldást fog ez a dokumentum a következőkben bemutatni.

2 VMware Vprobe

A VMware Vprobe egy diagnosztikai eszköz, ami a VMware Workstation 6.5-el együtt érkezett (már korábban is elérhető volt viszont ekkora érte el azt az érettségi fázist, hogy a kész termék mellé tegyék). Ez egy olyan eszköz amivel az egész virtualizált környezetet viszgálhatjuk. A következő képen látható az említett termék architektúrája.



1. ábra – Vmware

Tegyük fel hogy az egyes gépen Exchange 2000 fut Windows 2000 felett, a másodikon SQL Server Windows NT 4.0-en, a harmadikon pedig egy Apache egy Red Hat 9–es linuxon. Ebből már érzékelhető, hogy nem egyszerű feladat ezt a heterogén rendszert vizsgálni. A Vprobes különböző *probe*-okat definiál a rendszer különböző szintjein : a nyilakkal jelzett helyeken.

A megoldás lényege,hogy készíthetünk saját *probe*-okat, amiket a lehetséges pontokhoz csatlakoztathatunk. Ezek a probe-ok apró programok, amik képesek az azon a ponton elérhető adatokat begyűjteni. A *probe*-okat egy szkript nyelven készíthetjük el, a VPSript nyelven.

A *probe*-okat ahhoz hogy használni tudjuk, mindössze engedélyezni kell a Vmware Workstation programban és a figyelni kívánt virtuális gépen. Ezeket a megfelelő konfigurációs fájlokban tehetjük meg. Most pedig vegyük át, hogy milyen fő tulajdonságai vannak ennek a technolológiának.

2.1 A Vprobes tulajdonságai

A VProbes használata nem igényli a virtuális gép, se a hoszt gép ujraindítását, a hosztoló program újraindítását, vagy újrafordítását. Mint már korábban szerepelt, elég csupán a konfigurációs fájlokban engedélyezni a következő sorokkal:

Vmware Workstation : *vprobe.allow = TRUE*

Virtuális gép : *vprobe.enable = TRUE*

A *probe*-ok biztonságosak, mivel kizárólag adatot képesek gyűjteni, nem tudják módosítani a vizsgált környezetet, annak futását. A nyelv sem tartalmaz olyan elemeket amikkel olyan helyzetbe kerülhetnénk, amivel már akadályozzuk a működést .

Független a futtatott operácós rendszertől,mind a hoszton mind a virtuális gépeken. A *probe*-ok futtatása nem igényel számottevő erőforrásokat, így ezzel sem terheli le a gazda rendszert.

A rendszerre csatlakoztatható *probe*-okat több osztályba sorolhatjuk.

- statikus : Ezek az előre definiált pontok, mint például a virtuáls processzor egy „ütése” (VMM1Hz), egy guest hozzáférése egy virtuális hardverhez, Guest_IRQ…stb.
    - *(vprobe VMM1Hz*
      (printf "hello!\n"))

- dinamikus : Ezek saját magunk által definiálható csatolópontok. Ilyenek lehetnek például egy rendszerhívás a virtualizált rendszeren, egy memóriacím írása/olvasása.
    - *(vprobe GUEST:system_call*
      (printf "Hi from %s\n" PROBENAME))
    - *(vprobe GUEST:0xc0106ae0*
      (printf "reached 0xc0106ae0\n"))

Ha ki szereténk listázni az összes elérhető *probe* pontot, akkor futtassuk a következő parancsot:

```
vmrun vprobeListProbes my.vmx
```

2.2 *Probe*-ok készítése és használata

Probe-okat a már említett nyelven lehet készíteni. Miután elkészültünk egy *probe*-val és engedélyeztük a használatát, azután el kell indítani azt. Ezt UNIX rendszerek alatt a következő paranccsal tehetjük meg :

```
vmrun vprobeLoad my.vmx "`cat test.vp`"
```

Ez a parancs a *my.vmx* nevű virtuális gépre „akasztja” rá a *text.vp* nevű szkriptet.

Most pedig álljon itt egy kisebb példa *probe* :

```
; vptop.vp
;       Top-like VP script
; running is an aggregate that keeps a count of active process by name
; and which IRQ interrupted the process.
(defaggr running 1 1)
; Guest_IRQ is the probe that does the actual work
(vprobe Guest_IRQ
        (aggr running (ARG0) ((curprocname)) 1))
; Print and clear the aggregate once per second
(vprobe VMM1Hz
        (logaggr running)
        (clearaggr running))
```

Ez a kis probe egy *UNIX top „clone”* : minden egyes processzor ciklusban lefut és párosítja a az IRQ –t kérő alkalmazást a kéréssel. Ezután ezt egy aggregáló tárolóban helyezi el, amit aztán akár tovább is lehet irányítani más kimenetre is : konzol, fájl … stb.

Ehhez persze szükségesek saját segédfüggvények is mint például a cuprocname.

3 Tapasztalatok

A VMware Vprobe-ot használata sok szempontból könnyű ugyanakkor akadnak nehézségek is. Vegyük sorra őket.

3.1 Előnyök

- Ez egy rendkívül hatékony eszköz amivel virtualizált környezet működését lehet nyomon követni.
- Nem igényel semmiféle plusz telepítést, bonyolult konfigurációkat, extra jogosultságokat.
- Egyszerre akár több *probe*-t is akaszthatunk egy gépre, illetve több gépet is nyomon követhetünk párhuzamosan.
- A *probe*-ok függetlenek egymástól. Egyszerűek, nincs szükség terjedelmes fejlesztő környezetre, egy egyszerű szövegszerkesztő is elég.
- A szkript nyelv igen egyszerű, könnyű megtanulni. Ha ez nem elégíti ki az igényeinket – és lássuk be mivel igen egyszerű és alacsonyszintű a nyelv – a **VPSript** helyett használhatjuk az **Emett** nyelvet is amely azonban igencsak kísérleti fázisban van még, egy magasabb szintű nyelv minden előnyével és hátrányával.

A felsorolt tulajdonságok hozzájárulásával erre a „rétegre” építve készíthetünk saját diagnosztikai keretrendszert, amit alapvetően nem korlátoz se számítógéparchitektúra, se operációs rendszer. A korlátozás az, hogy a VMware terméket kell használnunk.

3.2 Hátrányok

- A nyelv egyszerűsége egyben hátrány is, mivel összetettebb *probe*-ok elkészítése igen idő- és munkaigényes lehet.
- Nincs hozzá fejlesztőkörnyezet, semmiféle grafikus felület.
- Csak VMware termékhez elérhető.
- Az operációs rendszerek mélyebb ismerete szükséges a használatához.
- Windows operációs rendszeren a *„command prompt”* hiányosságai miatt csak *Cygwin* alatt használható hatékonyan.

4 Összegzés

Összeségében igen hasznosnak találtam ezt az eszközt, sok érdekes dolgot lehet vele elvégezni.

(Mindazonáltal jövője kérdéseket vet fel, mivel az egyik fő fejlesztője **Keith Adams** elhagyta a VMware céget, illetve az termék hivatalos oldalán és fórumán igen kicsi volt az aktivitás az utóbbi időben).