Szolgáltatásbiztonságra tervezés laboratórium (VIMIM236)

Szolgáltatásbiztonsági benchmarkok

Mérési útmutató

Készítette: Kocsis Imre

2010.11.09.

Verzió: 1.0

Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

A mérés során két virtuális géppel fogunk dolgozni; egy Windows Server 2003 és egy CentOS 5 példánnyal. Ezek „bridged” VMWare hálózati módba vannak állítva; mivel a mérés során a hálózati interfészeket be és ki fogjuk kapcsolni, a DHCP-alapú IP-cím „lopások” elkerülése végett ajánlott a gépek első indításakor kapott IP címet fixre beállítani a virtuális gépekben. A virtuális gépekre belépéshez a root/LaborImage, illetve az Administrator/LaborImage felhasználói név/jelszó párosokat használjuk.

A mérésen az asztalokon található géppárokból az egyik hoszt lesz a *mért*, a másik pedig a *mérő* munkaállomás. Mindkét virtuális gép megtalálható mindkét gépen, a mérő gépen azonban csak a Windows Servert fogjuk használni (pontosabban az arra telepített JMetert). Mért gépnek javasoljuk a géppárok Lenovo tagját.

1 Ismerkedés a mérési környezettel

Mindkét virtuális gépen telepítésre került egy Tomcat szervlet-kiszolgáló, melyre egy klasszikus benchmark alkalmazást, az Apache Geronimo daytradert telepítettünk (korábban IBM Trade6).

I./1. A Tomcat egyik gépen sem lett telepítve szolgáltatásként. Keressük meg a Tomcat indítási mappáját (C:\tomcat\bin, illetve a /root-ban a megfelelő könyvtár) és konzolból próbáljuk ki a kiszolgáló elindítását és leállítását! (A startup, és shutdown szkriptek.)

I./2. A mérő gépről böngészőből próbáljuk ki és fedezzük fel a futó alkalmazást! Elérése: http://<IP cím>/daytrader. Az alkalmazás egy részvénykereskedési rendszert emulál; különlegessége, hogy szintetikus, tisztán benchmark jellegű hívásokhoz is biztosít URL-eket. Próbáljuk ezeket is ki! (A JDBCWrite és JDBCRead szintetikus hívások hibval térnek vissza; ennek az oka valószínűleg az, hogy alkalmazáskiszolgáló + valódi adatbázis-kiszolgáló helyett a munka megkönnyítése végett egy in-memory adatbázissal ellátott szervlet-konténeren fut az alkalmazás.)

I./3. A mérés során a mért virtuális gépekbe hibákat fogunk injektálni. VMWare környezetben ennek a legegyszerűbb módja a vmrun parancs használata a mért gazdahoszton. Ismerkedjünk meg a vmrun paranccsal (lásd www.vmware.com/pdf/vix160_vmrun_command.pdf, illetve az argumentumok nélkül futtatott parancs által kiírt dokumentációt) és próbáljunk meg a Windows vendég-gépen egy calc.exe-t elindítani, a Linuxon pedig egy új állományt touch-olni (lásd man touch)!

I./4. A mérés során a következő hibaokokat fogjuk injektálni (ezzel operátori, illetve környezeti hibákat emulálva): hálózati interfész kiesése, Tomcat folyamat leállása, illetve a „memory hog”, „I/O hog”, „CPU-hog” hármasból két szabadon választott.

- Tomcat kiesése: emulálható a shutdown és startup szkriptekkel.
- Hálózati interfész kiesése: próbáljuk ki az ifdown/ifup eth0, illetve a netsh interface set interface name="thename" admin=disable/enable parancsokat! (Praktikusan vmrun segítségével.)
- Az erőforrás-hibáknál választási lehetőséget biztosítunk a készen hozott alkalmazás és a „stress” nevű eszköz között. A stress által önmagáról adott dokumentáció a méréshez elengendő; a Windows Server hoszton cygwin alkalmazásként került telepítve a C:\cygwin\usr\local\bin\ könyvtárba, de cygwin bash indítása nélkül, egyszerű konzolos programként indítva is megfelelően működik.

2 Alapteljesítmény („baseline performance”) mérése

II./1. Készítsünk egy-egy snapshotot a virtuális gépek kiinduló állapotairól, amikhez majd rendre vissza tudunk térni!

II/2. Az alapteljesítmény méréséhez a mérő hoszton található Windows Serveren készítsünk egy terhelési profilt! A profil használhatja az oldal funkcionális részeit és a benchmark-hívásokat is, de

mindenképpen legyen indokolt a kérés sorozat felépítése! A terhelést méretezzük úgy kísérleti úton, hogy az mindkét mért virtuális gépen elfogadható teljesítményt adjon, de ne legyen triviálisan alacsony!

II./3. A mérésen az átlagos áteresztőképesség, az átlagos válaszidő és az átlagos hibaráta metrikákat szeretnénk benchmark metrikaként alkalmazni. Ezek méréséhez és rögzítéséhez egészítse ki megfelelő Listener-ekkel a terhelési profilt!

II./4. Nem túl nagy, de nem is nulla felfutási időt alkalmazva mérje meg az alapteljesítményt egy öt perces méréssel! (N.B. az itt elkészített profilt a hibainjektálások során módosítás nélkül fogjuk alkalmazni.)

3 Teljesítménymérések hibák jelenlétében

III./1. A mérési segédlet és a kiválasztott/előírt hibák alapján definiálja az injektálási réseket! Javasolt, hogy a hálózati interfész és a szervlet-kiszolgáló esetén ~40-50 másodpercenként legyen hibainjektálás (melyet ~5 másodperccel követ a jelen esetben automatikus helyreállítás), az erőforrás-foglaló megoldások esetén pedig ~30 másodpercenként úgy, hogy a hiba max. 10 másodpercig legyen aktív.

III/2. A hibák időzített injektálásához javasoljuk a következő perl szkript vázat:

```
#!/usr/bin/perl

$num = 5;
while($num--){

  @args = ("vmrun", "-gu", "Administrator", "-gp", "LaborImage",
  "runProgramInGuest", <path to .vmx>,
  "-noWait", "C:\\cygwin\\usr\\local\\bin\\stress.exe", "-c", "2",
  "-t", "10");
  system(@args) == 0
  or die "system @args failed: $?";
  sleep(20)
}
```

A sleep parancs az argumentumában megadott számú másodpercig felfüggeszti a szkript futását; a system hívás az argumentum-listája első elemeként megkapott parancsot hajtja végre a további listaelemek által meghatározott bemeneti paraméterekkel. A vmrun parancs noWait argumentuma használatára ahol kell, ügyeljen!

A szkriptet létrehozva és futtatási jogokkal ellátva (chmod +x) az a mért gazdagép termináljából egyszerűen indítható.

III./3. Hajtsa végre a hibainjektálási kampányt és rögzítse az eredményeket! Jelen mérés során a hibainjektálást és a terhelés elindítását nem szinkronizáljuk; a hibainjektálást indítsuk el „kézzel” a felfutási szakasz alatt! (Automatizált megoldásokat szívesen látunk, de nem követelmény.)

III./4. Értékeljük az eredményeket!