Szolgáltatásbiztonságra tervezés laboratórium (VIMIM236)

Szolgáltatásbiztonsági benchmarkok

Mérési segédlet

Készítette: Kocsis Imre

2013.11.20.

Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

1 Szolgáltatásbiztonsági benchmarkok

A *szolgáltatásbiztonsági benchmarkolás (dependability benchmarking)* olyan folyamat, melynek során egy rendszer szolgáltatásbiztonsági attribútumait (lásd [1]) értékeljük ki oly módon kísérletileg, hogy lehetővé váljon rendszerek összehasonlítása.

A különböző számítógépes rendszereknek a funkcionális követelmények mellett eleget kell tenniük *nemfunkcionális* (más szóhasználattal: *extrafunkcionális*) követelményeknek is. Alkalmazási területtől és konkrét alkalmazástól függ, hogy egy rendszer esetén mely szolgáltatásbiztonsági attribútum a legnagyobb jelentőségű; míg a különböző web-, illetve OLTP szolgáltatások esetén rendszerint a *rendelkezésreállás* és az *integritás* a legfontosabbak, addig a nem, vagy csak költségesen javítható, jellemzően beágyazott és/vagy missziókritikus rendszereknél (mosógéptől az űrszondáig) a *megbízhatóság* biztosítása az elsődleges cél. A hibás működés esetén potenciálisan életveszélyt okozó rendszerekben pedig a *biztonságosság* biztosítása élvez előnyt a többi attribútummal szemben.

A rendszertervezés során így érzékelhetően kulcsfontosságú, hogy a rendszer szolgáltatásbiztonsági aspektusairól valamilyen képet kapjunk; vagy a rendszer egészét értékelve, vagy az egyes komponenseket, melyek jellemzőiből vezetjük le a rendszerszintű tulajdonságokat. A tervezési folyamat során ezek a kiértékelések pl. akkor kapnak szerepet, amikor hasonló komponensek közül kell választanunk; pl. hogy egy több platformon is elérhető adatbázist milyen operációs rendszer felett futtassunk.

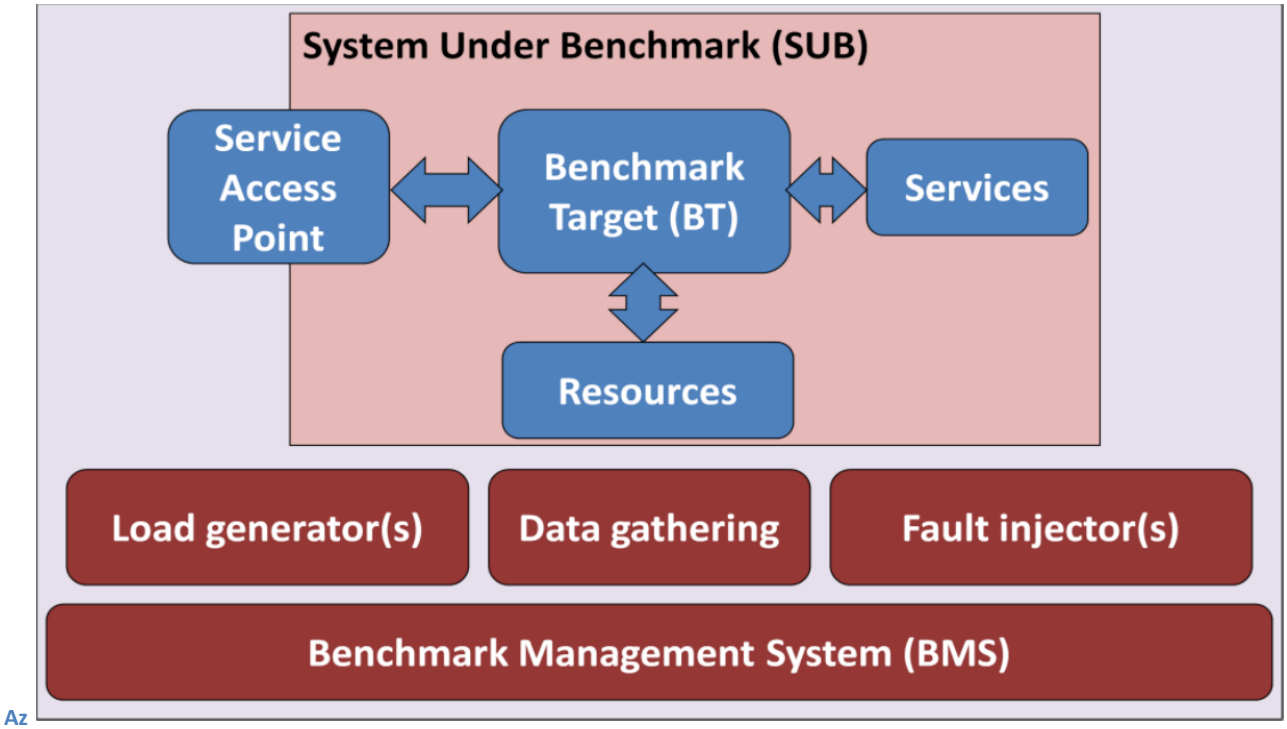
A tárgy laborgyakorlatai során több szolgáltatásbiztonság-kiértékelési megközelítéssel is megismerkedtünk már; ezek egy része formális modellekkel dolgozik (analitikus/szimuláció alapú), más része pedig alapvetően empirikus (pl. hibainjektálás, robosztusság-tesztelés). Észrevehettük, hogy e technikák igazi erejének kihasználásához szignifikáns mérnöki időráfordításra és szaktudásra van szükség. Míg missziókritikus (vagy az üzletmenet szempontjából kritikus) rendszerek esetén a rigorózus, az adott rendszerhez minél jobban illeszkedő szolgáltatásbiztonsági vizsgálatok indokoltak, az általános célú IT rendszerekhez ennél általánosabb megoldások szükségesek. Így a szolgáltatásbiztonsági benchmarkok célja az, hogy generikus keretet adjanak IT rendszerek viselkedésének leírására hibák jelenlétében, lehetővé téve a szolgáltatásbiztonsági tulajdonságok számszerűsítését.

1.1 Definíció

A szolgáltatásbiztonsági benchmarkok szorosan követik a teljesítménybenchmarkok filozófiáját; így a szolgáltatásbiztonsági benchmark fogalom definiálható úgy, mint egy szabványos folyamat specifikációja, mely egy számítógépes rendszer vagy rendszer-komponens szolgáltatásbiztonsági mértékeit értékeli ki [2]. Egy szolgáltatásbiztonsági benchmark főbb komponensei így a következők:

- **Munkaterhelés (workload)**: a benchmark futás során a rendszer által végzendő „munka”; pl. egy webes szolgáltatás és http terhelési profiljának definíciója.
- **Hibaterhelés (faultload):** hibák és aktiválásuk specifikációja; a hibaaktivációkkal létrehozott „hibaterhelés” célja, hogy a működés közbeni valós hibákat emulálja. (Bár, lévén a hibák általában ritka események, a valósnál jellemzően jóval nagyobb hibarátával.)
- **Mértékek (measures/metrics):** azon metrikák (és futásidejű kiértékelésük) definíciója, melyek a benchmarkolt rendszer teljesítményét és szolgáltatásbiztonsági tulajdonságait jellemzik.
- **Kísérleti elrendezés és a benchmark végrehajtásának folyamata.**

1.2 Kísérleti elrendezés szokásos elemei



Az

1. ábra bemutatja a kísérleti elrendezés szokásos főbb komponens-típusait és azok összefüggéseit.

System Under Benchmark (SUB)
Service Access Point
Benchmark Target (BT)
Services
Resources
Load generator(s)
Data gathering
Fault injector(s)
Benchmark Management System (BMS)

1. ábra. Szolgáltatásbiztonsági benchmarkok kísérleti elrendezésének szokásos főbb komponensei

1.3 Példák szolgáltatásbiztonsági benchmarkokra

A teljesség igénye nélkül megemlítünk néhány létező szolgáltatásbiztonsági benchmarkot. A DBench-OLTP [3] a TPC-C [6] OLTP adatbázis-benchmarkon alapul. [4] a Windows NT4, 2000 és XP szolgáltatásbiztonságát hasonlítja össze. Az IBM Autonomic Computing benchmark [5] egy, az IBM

Autonomic Maturity Model-en [7] alapuló kvalitatív „érettségi indexet” (maturity index) használ annak jellemzésére, hogy egy rendszer felügyelete „mennyire autonóm”. Az érettségi index szintjei:

Autonómia szintje	Leírás
Alapszint (basic)	Az IT komponensek menedzsmentje manuális, az egyes komponensek/termékek jelentési mechanizmusain alapul.
Menedzselt	Az IT taszkok végrehajtását és automatizálását felügyeleti szoftver támogatja.
Prediktív	Az egyes komponensek és rendszermenedzsment eszközök képesek a rendszer és környezete változásainak analizálásra és akciók javaslására.
Adaptív	Az IT komponensek kollektíven képesek a monitorozásra, kiértékelésre és akciók végrehajtására minimális emberi beavatkozással.
Autonóm	Az IT komponensek kollektív és automatikus menedzsmentjét magas szintű szabályok és eljárásrendek (policy-k) vezérlik.

1. táblázat. Az IBM Autnomic Maturity Model szintjei

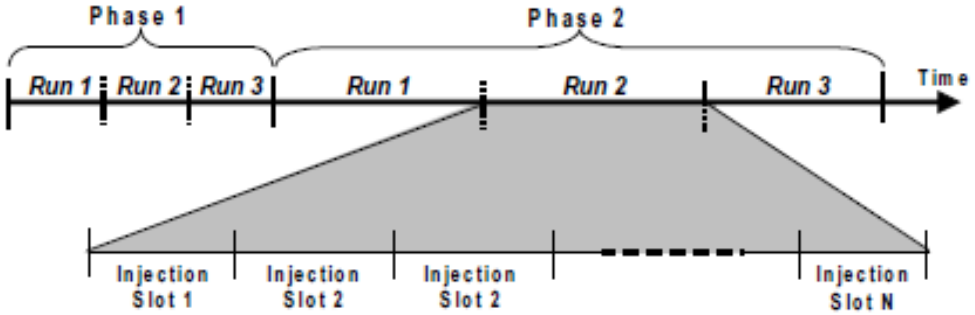
Mivel az operátori beavatkozások szükségessége (és hatékonysága) az IT rendszerek szolgáltatásbiztonságának alapvető faktorai, ezeket – bár más metrikákkal – más szolgáltatásbiztonsági benchmarkok is jellemzik.

2 Egy webkiszolgáló szolgáltatásbiztonsági benchmark

Ez a fejezet a [2] által javasolt webkiszolgáló szolgáltatásbiztonsági benchmarkot (WEB-DB) tekinti át röviden.

A WEB-DB benchmark újrahasznosítja a SPECweb99 [8] webszerver teljesítménybenchmark alapvető kísérleti elrendezését, munkaterhelését és teljesítmény-metrikáit; a benchmark futtatásának első fázisa valójában a SPECweb99 önmagában, hibaterhelés nélkül való futtatásából áll. Így előáll az *alapteljesítmény* (*baseline performance*), mely referenciaként fog szolgálni az aktív hibákkal zavart futásokkal szemben. A 2. fázisban a munkaterhelés már párhuzamosan fut a hibaterheléssel; azt vizsgáljuk, hogy a SUB-ba injektált hibák hogyan hatnak a szolgáltatás teljesítményére és szolgáltatásbiztonságára. Mind az első, mind a második fázis három azonos futás eredményeit átlagolja.

A második fázisban a benchmark a futásokat *injektálási résekre* (*injection slot*) osztja. Definíció szerint az injektálási rés egy olyan intervallum, ahol a rendszer munkaterhelést kap és emellett a hibaterhelés néhány hibáját injektáljuk.



2. ábra. A WEB-DB végrehajtási profiljának szerkezete

2.1 Metrikák

A WEB-DB futása eredményeként három különböző kategóriába sorolható metrikákat szolgáltat:

- *alapteljesítmény-metrikák*,
- *teljesítmény-metrikák hibák jelenlétében* és
- *szolgáltatásbiztonsági metrikák*.

Az 1. fázisban gyűjtött alapteljesítmény-metrikák:

- **SPEC**: a SPECweb99 fő metrikája, a szimultán „konform" kapcsolatok száma. Egy kapcsolat konform, ha a http kérések kiszolgálása során 1% százaléknál kevesebb a hiba és a kapcsolat átlagos bitrátája legalább 320 kbps.
- **THR**: áteresztőképesség, a másodperceként kiszolgált kérések száma.
- **RTM**: a kérések kiszolgálásának átlagos válaszideje.

A 2. fázisban a benchmark ugyanezen teljesítmény-metrikákat értékeli ki hibák jelenlétében, a megfelelő mértékeket rendre a következőképp jelölve: **SPECf**, **THRf**, **RTMf**.

A 2. fázisban történik meg a szolgáltatásbiztonsági metrikák gyűjtése és kiértékelése is.

- **Autonómia**: a webkiszolgáló „javításához" szükséges adminisztrátori beavatkozásokat jellemzi. Definíció szerint adminisztrátori beavatkozás akkor szükséges, ha a webkiszolgáló leáll, vagy az általa nyújtott szolgáltatás használhatatlanná válik. Számítása:

$$\text{Autonómia} = \left(1 - \frac{\#\ \text{adminisztr. beavatkozás}}{\#\ \text{hibák}}\right) * 100$$

- **Pontosság**: a válaszok hibarátáját adja meg hibaokok jelenlétében. Számítása:

$$\text{Pontosság} = \left(1 - \frac{\#\ \text{hibás kiszolgálású kérések}}{\#\ \text{kérések}}\right) * 100$$

- **Rendelkezésre állás**: a szokásos módon számítva a kérésekre adott helyes válaszok alapján és a futás teljes időtartamára vonatkoztatva.

2.2 Hibaterhelés

A hibaterhelés optimális esetben hibáknak és kivételes eseményeknek egy olyan halmaza, mely jól reprezentálja azon hibákat, melyek a webkiszolgálókat a gyakorlatban érintik. A megközelítés kritikusai szerint a szolgáltatásbiztonsági benchmarkok gyenge pontja pontosan a hibaterhelés reprezentativitása; pontosabban az, hogy szinte lehetetlen olyan hibaterhelést létrehozni, mely valóban általános érvényűen reprezentatív akár a vizsgált rendszerek, akár a lehetséges hibák halmazát tekintve. Ezen kritikák jogosak, ám – a teljesítménybenchmarkokhoz hasonlóan – a szolgáltatásbiztonsági benchmarkoknál jobb generikus eszköz nincs a kezünkben hasonló szolgáltatások szolgáltatásbiztonságának összehasonlítására.

Egy szolgáltatásbiztonsági benchmark hibaterhelése három alapvető hibaosztályból válogathat hibákat: *operátori hibák*, *szoftver hibák* és *hardver hibák*. A WEB-DB két különböző hibaterhelést alkalmaz: egy szoftver-hiba alapút és egy operátori hiba alapút, mely egyben a hardverhibákat is emulálja. A szoftver-hiba alapú hibainjektálással itt nem foglalkozunk, mivel

a mérés keretei nem teszik lehetővé ezek alkalmazását; megjegyzendő azonban, hogy a megfelelő technikák egy részét más kontextusban egy korábbi mérés már bemutatta.

A WEB-DB a következő operátori hibákat (és azok által emulált hardverhibákat) alkalmazza:

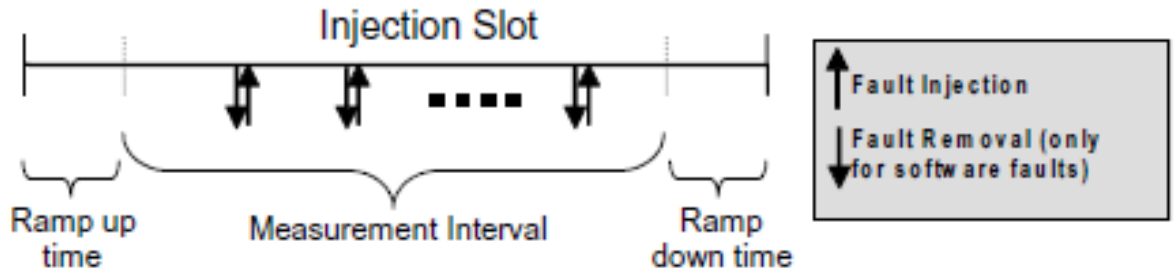
Hiba típus	Hibainjektálás módja
Hálózati interfész hiba	Hálózati kártya letiltása az operációs rendszerben.
Hálózati kapcsolat hibája	TCP socket-ek lezárása.
Hirtelen webkiszolgáló leállás	Webkiszolgáló folyamat leállítása.
Hirtelen kiszolgáló-újraindulás	Operációs rendszer újraindítása.

2. táblázat. A WEB-DB operátori hibái

A hibaterhelés ezeket a hibákat injektálja különböző jól definiált időpillanatokban.

2.3 Injektálási rések végrehajtási profiljai

A hibaterhelés teljes specifikációjához nem elég a hibákat kiválasztanunk; azokat injektálási résekhez kell rendelnünk és meg kell adnunk aktiválásuk időpontjait a réseken belül; emellett biztosítanunk kell, hogy a kísérletek megismételhetősége és az eredmények reprezentativitása teljesüljön.



3. ábra. Az injektálási rések szerkezete a WEB-DB-ben.

A WEB-DB esetén mindez a következők előírását jelenti.

1. A SUB állapotát minden injektálási rés elején explicit módon vissza kell állítani a kiinduló állapotba.
2. A munkaterhelés felfuttatása/a rendszer „bemelegedése", illetve a munkaterhelés lezárása miatt szükség van felfutási és lecsengési szakaszokra a rések elején és végén; ezen időszakok alatt ne történjen hibainjektálás. A SPECWeb99-ből adódóan a felfutás és a lecsengés 300 sec a WEB-DB esetén.
3. Az operátori hibákat egymástól elválasztva, külön-külön résben injektáljuk; az egyes 20 perces résekben ugyanazt a hibát többször is – a hálózati kapcsolati hibákat 20 másodpercenként, a hálózati kártya hibákat 40 másodpercenként, a webkiszolgáló-leállásokat 40 másodpercenként és a hoszt újraindításokat 10 percenként.
4. A hibák injektálása után egy diagnosztikai eljárást kell futtatni, mely eldönti, szükség van-e beavatkozásra a hiba helyreállításához. A helyreállítást a BMS vezérli, így emulálva az operátori javító akciókat.

3 A mérési feladatokról

A mérésen egy, a WEB-DB-hez hasonló szolgáltatásbiztonsági benchmark környezetet fogunk kialakítani és alkalmazni. A környezet főbb ismérvei a következők.

- Terhelésgenerálásra és metrikák gyűjtésére, kiértékelésére az egyik korábbi mérés során már megismert Apache JMetert alkalmazzuk, így az azzal kapcsolatos ismereteket érdemes felelevéníteni.
- Szolgáltatásként egy egyszerű Java EE mintaalkalmazást használunk, virtuális gépben futtatva.
- A hibainjektálások automatizálását és ütemezését, valamint a virtuális gép snapshothoz való visszatérést egy általunk biztosított alkalmazás támogatja.
- A hibainjektor-készlet nagy részét mi biztosítjuk.

A mérés során ugyanazon szolgáltatás különböző operációs rendszerek alatti viselkedését fogjuk vizsgálni.

4 Felkészülés a mérésre

A mérésre való felkészülés része két környezeti hibainjektor Java alkalmazás elkészítése a következő három opció közül: „CPU-hog”, „Memory-hog” és „diszk I/O-hog”. (Azaz az egyes alkalmazásoknak ezeket az erőforrásokat kell minél jobban kiterhelniük.) Ügyeljen arra, hogy az alkalmazásnak mind Linux, mind Windows rendszerek felett tudnia kell futni és a kívánt hatást elérni! A hibainjektorok tervezési döntéseinek dokumentációja a mérési jegyzőkönyv részét kell, hogy képezze.

5 Hivatkozások

[1] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, “Basic concepts and taxonomy of dependable and secure computing,” *IEEE Transactions on Dependable and Secure Computing*, vol. 1, Jan. 2004, pp. 11-33.

[2] J. Durães, M. Vieira, and H. Madeira, “Dependability benchmarking of web-servers,” *Computer Safety, Reliability, and Security*, Springer Berlin / Heidelberg, 2004, pp. 297-310.

[3] M. Vieira, “A dependability benchmark for OLTP application environments,” *of the 29th international conference on Very large data bases*, 2003, p. 753.

[4] A. Kalakech, K. Kanoun, Y. Crouzet, and J. Arlat, “Benchmarking the dependability of windows NT4, 2000 and XP,” *International Conference on Dependable Systems and Networks, 2004*, IEEE, 2004, pp. 632-637.

[5] J. Coleman, T. Lau, B. Lokhande, P. Shum, R. Wisniewski, and M.P. Yost, “The autonomic computing benchmark,” *Dependability Benchmarking for Computer Systems*, IEEE Computer Society, 2008, pp. 1-21.

[6] TPC-C: http://www.tpc.org/tpcc/

[7] IBM Corporation, “Understand autonomic maturity levels,” http://www.ibm.com/developerworks/library/ac-mature.html

[8] SPECweb99: http://www.spec.org/web99/