Kommunikáló rendszerek teljesítménytesztelése

ERŐS LEVENTE
BME Távközlési és Médiainformatikai Tanszék
eros@tmit.bme.hu
PERNEK ÁKOS
BME Automatizálási és Alkalmazott Informatikai Tanszék
akos.pernek@aut.bme.hu
CSÖNDES TIBOR
Ericsson Magyarország Kft., Test Competence Center
tibor.csondes@ericsson.com

Kulcsszavak: teljesítménytesztelés, teljesítménymodellezés, tesztrendszerek

Cikkünkben bemutatjuk a teljesítménytesztelés néhány alapvető módszerét és egy olyan feketedoboz-alapú terheléstesztelési eljárást, amellyel automatikusan mérhetjük ki kommunikáló rendszerek teljesítményét, és amely kiküszöböli az ad-hoc terheléstesztelés okozta problémákat. Az eljárás kétfázisú: első fázisként a teljesítménykövetelményeket automatikusan képezzük le egy teljesítménymodellre, majd a második fázisban ezt a modellt hasonlítjuk össze a valós tesztelt rendszerrel.
Ez utóbbi fázisban történő mérések eredményét az eljárás kiértékeli, és megállapítja, hogy a tesztelt rendszer legrosszabb esetben (azaz bármilyen bemeneti üzenetsorozat esetén) valamint átlagosan hogyan fog teljesíteni. A cikk végén bemutatunk néhány szimulációs eredményt az eljárás hatékonyságának alátámasztására.

1. Bevezetés

A tesztelés a minőségbiztosítás egy fontos állomása mind a hardver, mind a szoftveriparban. Ebben a cikkben a számos tesztelési módszer közül a terheléstesztelés módszere illetve ennek különböző metodológiái kerülnek bemutatásra.

A terheléstesztelés egy teljesítménytesztelési módszer. A teljesítménytesztelés egy általános tesztelési kategória, mely során a tesztelés célja a vizsgált rendszer karakterisztikájának meghatározása különböző mértékű terhelések alatt. A teljesítménytesztelés kategóriába tartozik, például az úgynevezett stressztesztelés, illetve a „benchmark” tesztelés. Az előbbi esetben a cél a rendszer extrém túlterhelése illetve annak vizsgálata, hogy a rendszer megfelelően képes-e újraéledni, míg az utóbbi esetben a rendszer terhelése minimális. A cél a rendszer alapvető funkcióinak vizsgálata megfelelő működési, használhatósági és egyéb szempontok alapján.

A terheléstesztelés során a rendszer terhelése magas, azonban ez a terhelés nem túlterhelés, hanem a bizonyos rögzített körülmények mellett elvárható maximum. A rögzített körülmények jelenthetik a szimulált felhasználók maximális számát, a párhuzamosan futó tranzakciók maximális számát és sok egyéb speciális körülményt. A terheléstesztelés során ezen megszorítások mellett kell a tesztelt rendszert vizsgálni. A vizsgálat eredményeképpen megállapítható, hogy a rendszer képes-e teljesíteni az elvárt teljesítménykövetelményeket, továbbá mérhetőek a pontos teljesítménymetrikái is.

2. A terheléstesztelés módszerei

A terheléstesztelés során is alkalmazott egyik módszer az úgynevezett feketedoboz-tesztelés. A feketedoboz-tesztelés során a rendszer belső felépítése nem ismert, a tesztek a rendszer külső specifikációja alapján kerülnek kialakításra. Egy másik – a feketedoboz-alapú teszteléssel ellentétes filozófián alapuló – gyakran alkalmazott módszer a fehérdoboz-alapú tesztelés. Fehérdoboz-alapú tesztelés esetén a tesztelt rendszer belső tulajdonságai ismertek. A tesztek létrehozása az ismert belső tulajdonságok alapján történik.

Míg a feketedoboz-alapú tesztelés célja a tesztelt rendszer teljesítménymutatóinak vizsgálata, mérése és egyben a rendszer verifikációja abból a szempontból, hogy a specifikált teljesítményt valóban tartani képes-e, addig a fehérdoboz-alapú teljesítménytesztelés célja a rendszer belső működésének, teljesítményének vizsgálata. Az ismert belső felépítésű rendszer viselkedést figyelve szakaszosan növekedő terhelés mellett azonosíthatóvá válnak a rendszer túlterhelt pontjai mind kód, adatbázis, rendszer és hálózati szinten. A szűk keresztmetszetért felelős egységek optimalizálásra kerülhetnek.

Mindkét módszer fontos követelménye, hogy a rendszer funkcionálisan megbízható legyen, vagyis a terheléstesztelés előtt a rendszernek konformanciatesztelésen kell átesnie. A feketedoboz-alapú konformanciateszt során azt vizsgáljuk, hogy a tesztelt rendszer a megfelelő kommunikációs protokollt valósítja-e meg, azaz, hogy a megfelelő bemeneti sorozatokra a megfelelő kimeneti sorozatokat küldi-e vissza az időzítéseket is figyelembe véve. Fehérdoboz-alapú tesztelés során a konforman-

cia tesztek bizonyosságot adhatnak arról, hogy a tesztelt rendszer belső működése is az elvártnak megfelelő. A teljesítménykövetelmények teljesülését vizsgáló terheléstesztelésre akkor kerülhet sor, ha a rendszer megfelelt a konformanciateszten. A konformanciatesztelés mára kiforrott elméleti háttérrel rendelkezik [1-4].

3. Tesztrendszerek napjainkban

A terheléstesztelésnek alávetett kommunikációs rendszerek közös tulajdonsága, hogy van legalább egy interfészük. Az interfészek biztosítják a kapcsolatot a külvilág felé. Minden interfész implementál egy adott protokollt. A tesztelt kommunikáló rendszerek további fontos paraméterei közé tartozik a szimulált felhasználók maximális száma illetve a párhuzamosan végrehajtott tranzakciók maximális száma.

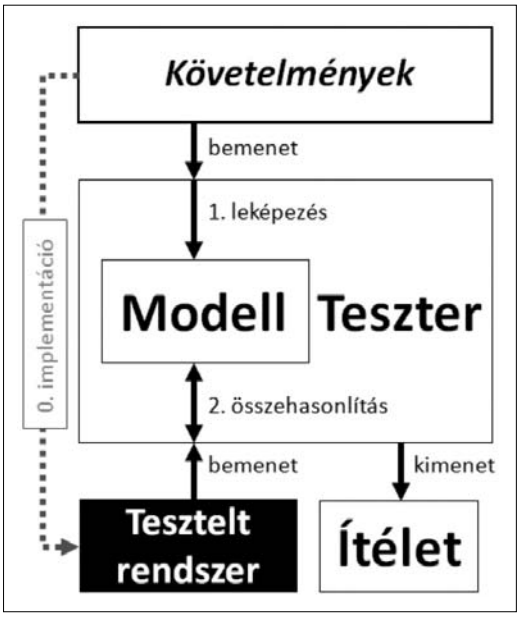
A fenti és esetlegesen egyéb tulajdonságok erős befolyást gyakorolnak az adott eszköz tesztelését megvalósító tesztrendszerre is. A terheléstesztelés során a használt tesztrendszernek képesnek kell lenni a tesztelt rendszer teljesítményének megfelelő terhelést hosszú ideig és megbízhatóan előállítani.

A tesztrendszernek a tesztelés során le kell fednie a tesztelt rendszer aktív interfészeit. Az interfészek lefedése manapság leggyakrabban szoftverkomponensek révén történik. Ennek legfőbb oka a költséghatékonyság. Az interfészek lefedése történhet speciálisan legyártott célhardverrel is, azonban ezen eszközökre jellemző a kis darabszámban történő gyártás és ennek következtében az igen magas gyártási és értékesítési költségek. A személyi számítógépek alkalmazása hatalmas költségmegtakarítást eredményez, illetve alkalmazhatóságuk és skálázhatóságuk révén az általuk biztosított teljesítmény is meghaladja a célhardverek képességeit. A célhardverek alkalmazásának csökkenése, illetve csökkentése azonban nem egyértelmű folyamat. A hagyományos telekommunikációs világban a tesztelt rendszerek környezetének szimulálása során szükséges volt bonyolult és költséges célhardverek alkalmazása. Azonban a távközlési világ elmozdulása az IP-s világ irányába egyre kevésbé tette szükségessé ezen a hardverek alkalmazását.

A személyi számítógépek alkalmazásának másik – korábban már részben említett – előnye a skálázhatóság. Egy megfelelően megtervezett tesztrendszer képes kihasználni több számítógép erőforrásait is, mely révén a teljesítmény növelése egyszerűen és nem utolsó sorban a célhardverekhez viszonyítva továbbra is olcsón növelhető. Az esetek jó részében azonban egy számítógép is képes a szükséges tesztelési teljesítmény előállítására. Napjaink „boltban” megvásárolható személyi számítógépeinek kapacitása a lehetséges maximális szimulált felhasználószámot is nagyban növelheti. Egy tesztelés alatt álló rendszer esetében az elvárt maximális felhasználószám gyakran több milliós populációt jelent, azonban az esetek többségében ez már nem jelent problémát.

Egy modern tesztrendszer gyakran modellvezérelt, mely modell a tesztelés alatt álló rendszer viselkedésspecifikációja alapján kerül kialakításra. A modell által vezérelt rendszerek bővíthetősége egyszerű, könnyű őket felkészíteni mind a sikeres, mind a sikertelen lefolyású tesztesetekre, illetve gyakran negatív tesztelésre is alkalmazhatóak. A tesztelt rendszer karakterisztikájának mérése miatt fontos, hogy a használt tesztrendszer stabilan képes legyen leadni a szükséges teljesítményt, ezért célszerű a tesztrendszert felkészíteni a sikertelen lefolyású tesztesetekre is. A robusztus tesztrendszer elengedhetetlen követelmény a megfelelő terheléstesztelés szempontjából.

Végezetül megemlítjük, hogy a legtöbb tesztrendszer rendelkezik grafikus felhasználó felülettel, mely felület lehetővé teszi a tesztek egyszerű vezérlését, akár személyre szabását is, valamint a teszt aktuális állapotának megfigyelését és a teszt eredményeinek elmentését a későbbi elemzés céljából.



1. ábra Terheléstesztelési eljárás

4. Ad-hoc- és modellalapú teljesítménytesztelés

A terheléstesztet manapság főként ad-hoc módon, kézzel implementálják, a tesztmérnökök korábbi tapasztalataira támaszkodva. Ad-hoc-tesztelés során az implementált tesztrendszer terhelés alá helyezi a tesztelt eszközt és a rendszer válasza alapján próbálja meghatározni a tesztelt rendszer valódi paramétereit. Ennek a módszernek a nyilvánvaló hátránya a mérések pontatlansága.

Cikkünkben ezen probléma megoldásaként egy automatikus terheléstesztelési módszert mutatunk be, amely elsőként leképez két teljesítménykövetelményt, méghozzá a másodpercenként feldolgozott üzenetszámot és a párhuzamosan kiszolgált felhasználók számát egy formális modellre, majd ellenőrzi, hogy a tesztelt rendszer megfelel-e ennek a modellnek *(1. ábra)*.

A teljesítménymodellezés témakörében már számos publikáció született [5-8]. Ezek azonban a modell *verifikálását* tűzik ki céljukként, azaz annak analitikus bizonyítását, hogy a modellek megfelelnek az előírt teljesítménykövetelményeknek. A mi célunk ezzel szemben az volt, hogy egy rendszer *validálására* adjunk eljárást, azaz, hogy eldöntsük, hogy egy fizikai rendszer megfelel-e a teljesítménykövetelményeknek.

5. A teljesítménykövetelmények leképezése formális teljesítménymodellre

Ebben a szakaszban bevezetjük az időzített kommunikáló véges többállapotú gép (Timed Communicating Finite Multistate Machine, TCFMM) modellt, majd megvizsgáljuk, hogy segítségével hogyan lehet modellezni a tesztelt rendszer által másodpercenként feldolgozott üzenetek számát valamint a párhuzamosan kiszolgált felhasználók számát, mint teljesítménykövetelményt. A TCFMM-modellt a következő attribútumok írják le:

$$TCFMM = (S,I,O,T,U,s_0),$$ ahol

S az állapotok véges halmaza, *I* a bemenetek véges halmaza, *O* a kimenetek véges halmaza, *T* az állapotátmenetek (tranzíciók) véges halmaza, *U* a felhasználók (tokenek) véges halmaza, végül s_0 a rendszer kezdőállapota, ahol kezdetben a felhasználókat reprezentáló tokenek tartózkodnak. Egy állapotátmenethez tartozik egy bemenet, egy kimenet valamint egy késleltetés. Ha a rendszer valamely felhasználótól olyan üzenetet kap, amely a felhasználó tokenjének aktuális állapotából kiinduló valamely állapotátmenet bemenete, akkor a tokent elveszi az aktuális állapotból, az állapotátmenethez tartozó késleltetés leteltét követően elhelyezi annak célállapotában és az állapotátmenethez tartozó kimenetet válaszüzenetként elküldi a felhasználónak.

A tesztelt rendszer modellezéséhez használt TCFMM struktúráját a tesztelt rendszer által megvalósított protokollhoz tartozó kommunikáló kiterjesztett véges állapotautomata adja. Ezen állapotautomata állapotátmeneteit a teljesítménymodell megalkotásához ki kell egészíteni egy-egy késleltetésparaméterrel (amely értékét még nem ismerjük), és az automata kezdőállapotába el kell helyezni az |U| darab tokent. A *2. ábrán* látható két TCFMM, amely állapotátmenetein *bemenet/kimenet/késleltetés* formában jelenik meg az egyes paraméterek értéke.

Ahhoz, hogy a fentiekben leírt módszerrel megalkotott TCFMM megfelelően modellezze a párhuzamosan kiszolgálandó felhasználók maximális számát, pontosan annyi tokent kell elhelyezni a modell kezdőállapotában, ahány felhasználót a rendszernek egyszerre ki kell szolgálnia. Ezen kívül azt is meg kell oldanunk, hogy valahányszor egy felhasználó egy nyelőállapotba viszi a hozzá tartozó tokent (tehát a token olyan állapotba kerül, amelyből nem vezet ki állapotátmenet, és így a felhasználó nem generálhat több kérést a rendszer felé, a rendszer szemszögéből tehát a kiszolgált felhasználók száma eggyel csökken), egy új token, azaz egy új felhasználó kiszolgálásának lehetősége jelenjen meg a kezdőállapotban. Ezt úgy érjük el, hogy valamennyi, nyelőállapotba vezető állapotátmenetet átirányítunk a kezdőállapotba. Az 1. ábra jobb oldalán látható TCFMM úgy keletkezett, hogy a bal oldali TCFMM egyetlen nyelőállapotába mutató állapotátmeneteket átirányítottuk a kezdőállapotba.

A rendszer által másodpercenként feldolgozott üzenetek számát az állapotátmenet-késleltetésekre felállított megkötésekkel modellezzük. Itt kell megemlítenünk, hogy a másodpercenként feldolgozott üzenetszám alatt a rendszer által hosszú (optimálisan végtelen) idő alatt feldolgozott üzenetek számának és a mérés másodpercben kifejezett idejének hányadosát értjük, azaz egy hosszú időre mért átlagot. Első megközelítésben azt mondhatjuk, hogy a tesztelt rendszer akkor képes másodpercenként *C* darab üzenetet feldolgozni, ha valamennyi állapotátmenetének másodpercben kifejezett késleltetése kisebb, mint *C* reciproka, azaz, ha a t_i állapotátmenet késleltetését d_i-vel jelöljük; $d_i \le \frac{1}{C}$.

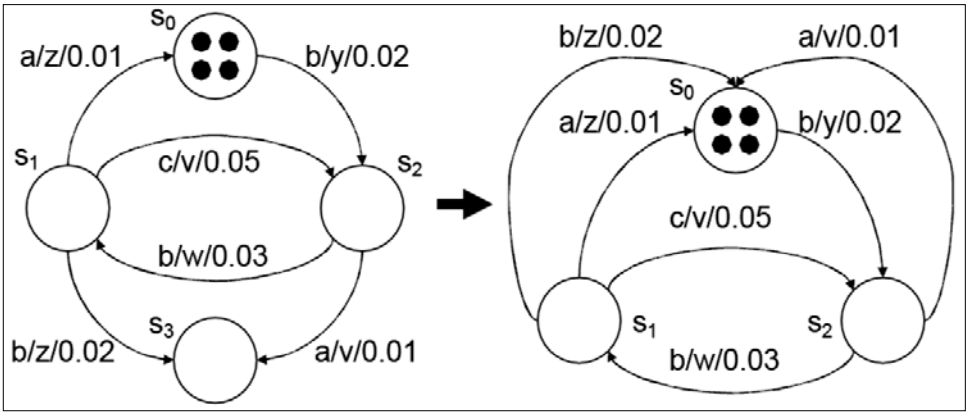
Ezen követelmény túl szigorú, ugyanis ha egy állapotátmenet késleltetése nem teljesíti, a rendszer hosszútávon még mindig képes lehet *C* üzenet kiszolgálására másodpercenként, méghozzá valamennyi állapotátmeneti trajektória mentén. Ennek feltétele, hogy a „lassú" állapotátmenet késleltetését kompenzálják azon állapotátmenetek késleltetései, amellyel ez az állapotátmenet ugyanazon kör(ök)ben szerepel a TCFMM-modellben.

Formálisan tehát a rendszer képes másodpercenként *C* üzenetet feldolgozni, ha $\frac{\sum_{t_j \in c_i} d_j}{|c_i|} \le \frac{1}{C_{req}}$,

ahol c_i egy kört jelöl és ez a feltétel valamennyi körre teljesül.

Belátható, hogy ez a követelmény már szükséges és elégséges feltétele annak, hogy a tesztelt rendszer bármilyen bemeneti sorozat esetén képes legyen a másodpercenkénti *C* darab üzenet feldolgozására.

2. ábra TCFMM-modellek



6. Teljesítménymodell összehasonlítása a tesztelt rendszerrel

Miután modelleztük a rendszerrel szemben felállított teljesítménykövetelményeket, ki kell mérnünk a rendszer teljesítményét. Precízebben megfogalmazva, azt kell kiszámolnunk, hogy a tesztelt rendszer másodpercenként hány üzenetet dolgoz fel, miközben az általa kiszolgált felhasználók száma rögzítetten a maximális felhasználószám. Ehhez minden egyes állapotátmenet késleltetését kell kimérnünk (tehát azt az időtartamot, amely a tesztezköz által a tesztelt rendszernek küldött bemeneti üzenet elküldése és a válaszüzenet vétele között telt el).

A késleltetések birtokában kétféleképpen is kiszámolhatjuk a rendszer által másodpercenként feldolgozott üzenetek számát. A *legrosszabb,* azaz minimális másodpercenkénti üzenetszámot a következő képlettel kaphatjuk meg:

$$C_{\min} = \min\left\{\frac{|c_j|}{\sum_{t_i \in c_j} d_i}\right\}$$

A fenti képlet szerint tehát meg kell keresni azt a kört a TCFMM-ben, amelynek az egy tranzícióra eső átlagkésleltetése maximális. A keresett minimális tüzelésszám az ezen kör mentén mért tüzelésszám lesz. Ez egybecseng az előző szakaszban az állapotátmenet-késleltetésekre megszabott megkötéssel.

Ha a teszt során nem arra vagyunk kíváncsiak, hogy *legalább* hány üzenetet dolgoz fel a rendszer egy másodperc alatt, hanem arra, hogy *átlagosan* hányat, a minimális üzenetszám helyett, amely egy meglehetősen szigorú becslés a rendszer teljesítményére, megbecsülhetjük az átlagosan feldogozott másodpercenkénti üzenetszámot. Ennek kiszámításához azonban szükség van egy kis pluszinformációra a tesztelt rendszer jövőbeni felhasználóinak viselkedéséről. Jelölje q_i annak valószínűségét, hogy egy token a t_i tranzíció forrásállapotában állva a t_i tranzíció bemenetét küldi a tesztelt rendszernek, azaz a t_i tranzíciót tüzeli el (a q_i értékekből megalkothatjuk tehát a felhasználók viselkedését leíró véges Markov-láncot). Jelölje továbbá p_{kl} azon tranzíciók q_i valószínűségeinek összegét, amelyek s_k-ból s_l állapotba mennek.

Ezek és a kimért késleltetésértékek alapján első lépésben ki kell számítanunk a felhasználók viselkedését leíró Markov-lánc stacioner eloszlását. Az *i*-edik állapot stacionárius állapotvalószínűségét z_i-vel jelöljük. Ezt a következő mátrixegyenlet megoldásával határozhatjuk meg:

$$\underline{\underline{F}}\,\underline{z} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \text{ ahol } \underline{z} = \begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ z_n \end{bmatrix} \text{ és } \underline{\underline{F}} = \begin{bmatrix} p_{00}-1 & p_{10} & \cdots & p_{n0} \\ p_{01} & p_{11}-1 & \cdots & p_{n1} \\ \vdots & \vdots & \ddots & \vdots \\ p_{0(n-1)} & p_{1(n-1)} & \cdots & p_{n(n-1)} \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

Az egyenletnek akkor lesz megoldása, ha $\det \underline{\underline{F}} \neq 0$ [9]. A stacioner állapotvalószínűségek ismeretében megadhatjuk valamennyi állapotátmenet stacioner tüzelési valószínűségét, amely annak valószínűsége, hogy egy felhasználó mekkora valószínűséggel tüzeli éppen az adott állapotátmenetet. Az *i*-edik tranzíció stacioner tüzelési valószínűsége: $f_i = z_k q_i$. Ennek alapján az átlagos másodpercenkénti üzenetszámot a következőképpen számíthatjuk:

$$C_{\text{átlag}} = \frac{1}{\sum_{t_i \in T} d_i f_i}$$

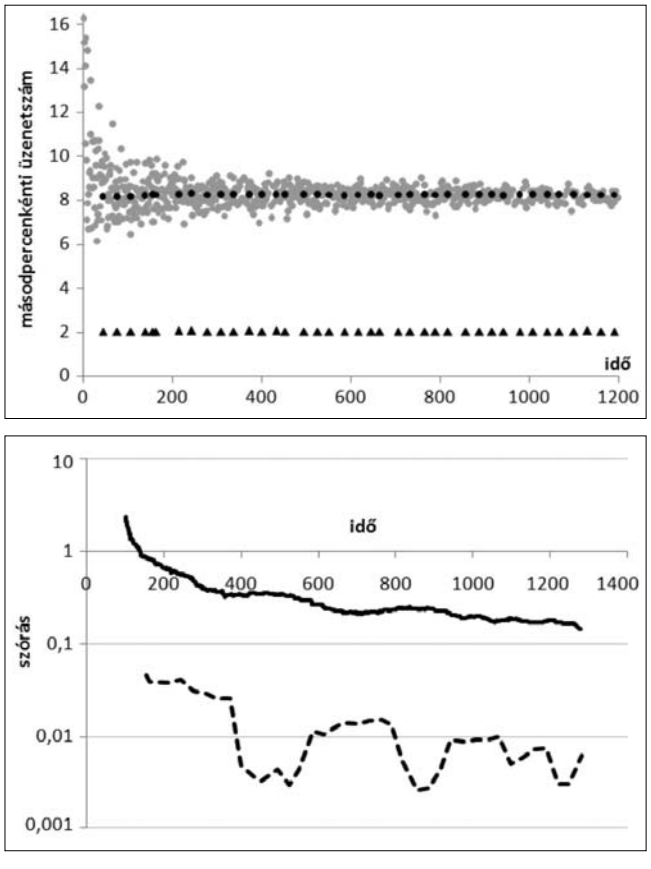
A fenti képlet szerint tehát venni kell a tranzíciók késleltetéseinek stacionárius tüzelési valószínűségeikkel súlyozott átlagát és az így kapott érték reciproka adja az átlagos tüzelésszámot.

7. Szimulációs eredmények

Ebben a szakaszban bemutatunk néhány szimulációs eredményt az előző szakaszokban leírt módszerek hatékonyságának szemléltetésére. A bemutatott eljárást egy ad-hoc terheléstesztelési módszerrel hasonlítottuk össze, amely során a felhasználók viselkedését teszterentitások emulálják és azt mérik, hogy a teszt futásának teljes ideje alatt hány üzenetet szolgált ki feléjük a tesztelt rendszer.

A *3. ábrán* az ad-hoc, illetve a cikkben bemutatott módszerrel ugyanazon a rendszeren mért másodpercenkénti üzenetszámértékek láthatóak a méréshez szükséges idő függvényében. Látszik, hogy a minimális másodpercenkénti üzenetszám (fekete háromszögek) egy elég durva alsó becslés a rendszer teljesítményére, illetve az is, hogy az itt leírt módszerrel jóval precízebben mérhető

3-4. ábra Mérési eredmények és azok szórása



ki a tesztelt rendszer által feldolgozott átlagos üzenetszám (fekete körök), mint az ad-hoc módszerrel (szürke körök).

Ezt támasztja alá a *4. ábra* is, amelyen az ad-hoc (folytonos vonal), illetve a cikkben bemutatott módszerrel (szaggatott vonal) végzett teljesítménymérések eredményeinek szórása látszik a méréshez szükséges idő függvényében, logaritmikus skálán. Ennek alapján az utóbbi módszer mérési eredményeinek szórása két nagyságrenddel alacsonyabb az ad-hoc módszerénél.

8. Összefoglalás

Cikkünkben bemutattunk a teljesítménytesztelés főbb módszereit, illetve egy olyan teljesítménytesztelési eljárást, amely a feketedoboz-alapú terheléstesztelés ad-hoc voltából adódó problémákat oldja meg és a teljesítménykövetelményektől automatikusan jut el a tesztelt rendszer teljesítményének kiméréséig.

Az automatikus működést az teszi lehetővé a bevett gyakorlattal ellentétben, hogy a tesztmérnök tapasztalatai helyett egy formális teljesítménymodellre támaszkodik, amelyre első lépésben leképezi a rendszerrel szemben támasztott teljesítménykövetelményeket, esetünkben a párhuzamosan kiszolgálandó felhasználók maximális számát és a rendszer által másodpercenként feldolgozandó üzenetszámot, majd a rendszer állapotátmeneti késleltetéseinek a modell alapján történő kimérését követően megállapítja, hogy a rendszer legalább, illetve várhatóan hány üzenetet lesz képes feldolgozni másodpercenként. Az elvégzett szimulációk igazolják a bemutatott módszer hatékonyságát és precizitását.

A szerzőkről

ERŐS LEVENTE 2007-ben szerzett mérnök-informatikusi diplomát a Budapesti Műszaki és Gazdaságtudományi Egyetem Villamosmérnöki és Informatikai Karán. 2007 óta a Kar Távközlési és Médiainformatikai Tanszékének doktorandusz hallgatójaként végez kutatómunkát. Kutatási területe kommunikáló rendszerek fekete doboz alapú teljesítmény-tesztelése.

PERNEK ÁKOS 2007-ben végzett a Budapesti Műszaki és Gazdaságtudományi Egyetem Villamosmérnöki és Informatikai Karán. 2007 óta az Automatizálási és Alkalmazott Informatikai Tanszék doktorandusz hallgatója. 2006-tól az Ericsson Magyarország munkatársa TTCN-3 tesztrendszer fejlesztői munkakörben. A Magyar Tudományos Akadémia Számítástechnikai és Automatizálási Kutatóintézetének munkatársa. Főbb kutatási területei: kamera-autokalibráció és pontos objektumrekonstrukció videó alapján.

CSÖNDES TIBOR 1996-ban végzett a Budapesti Műszaki Egyetem Villamosmérnöki és Informatikai Karán. 1996-tól 1999-ig a kar doktorandusz hallgatója. PhD disszertációját 2002-ben védte meg konformancia tesztsorozatok optimalizálása témakörben. 1997-től az Ericsson Magyarország munkatársa; kutató, majd tesztelési rendszermérnök, 2009-től osztályvezetői munkakörben. Az Ericsson hivatalos nemzetközi TTCN-3 tanfolyamának előadója. A BME Villamosmérnöki és Informatikai Karának címzetes egyetemi docense. Több diplomamunka és PhD téma konzulense valamint a Távközlési Szoftverek című doktorandusz tárgy társelőadója. Főbb kutatási és oktatási területei: konformancia tesztelés, tesztsorozat optimalizálás, TTCN-3 tesztleíró nyelv, automatikus tesztelés, tesztgeneráló módszerek, modell alapú tesztelés.

Irodalom

[1] ISO/IEC 9646:
Information technology
– Open Systems Interconnection – Conformance testing methodology and framework, 1994.

[2] C. Feng, X. Sun, Y. Shen F. Lombardi,
Protocol Conformance Testing
Using Unique Input/Output Sequences.
World Scienific, 1997.

[3] C. Kim, J. Song,
Test Sequence Generation Methods for
Protocol Conformance Testing,
In: Proc. of the 18th Annual International Computer Software and Applications Conf., pp.169–174., 1994.

[4] ITU-T, Framework on Formal Methods in Conformance Testing, ITU-T Recommendation Z.500.
Geneva, Switzerland, 1997.

[5] P. Kemper, P. Kritzinger, F. Bause, H. Kabutz,
SDL and Petri Net Performance Analysis of Communicating Systems,
In: Proc. of the 15th International Symposium on Protocol Specification, Testing and Verification.
Chapman and Hall, pp.269–282., 1995.

[6] O.S. Youness, W.S. El-Kilani, W.F.A. El-Wahed,
A Behavior and Delay Equivalent Petri Net Model for Performance Evaluation of Communication Protocols,
Computer Communications,
Vol. 31, No. 10, pp.2210–2230., 2008.

[7] M.R. El-Karaksy, A.S. Nouh, A. Al-Obaidan,
Performance Analysis of Timed Petri Net Models for Communication Protocols: a Methodology and Package,
Computer Communications,
Vol. 13, No. 2, pp.73–82., 1990.

[8] M.A. Marsan, G. Chiola, A. Fumagalli,
Timed Petri Net Model for The Accurate Performance Analysis of CSMA/CD Bus Lans.
Computer Communications,
Vol. 10, No. 6, pp.304–312., 1987.
http://dblp.uni-trier.de/db/journals/comcom/comcom10.html#MarsanCF87

[9] K. Hoffman, R. Kunze,
Linear Algebra (2nd ed.),
Prentice Hall, 1971.