Budapesti Műszaki és Gazdaságtudományi Egyetem
Műszaki Tudományág, Informatikai Tudományszak

WEBALAPÚ SZOFTVERRENDSZEREK TOVÁBBFEJLESZTETT TELJESÍTMÉNYMODELLJEI

IMPROVED PERFORMANCE MODELS FOR WEB-BASED SOFTWARE SYSTEMS

Ph.D. értekezés tézisei

Bogárdi-Mészöly Ágnes

Tudományos vezetők:
Dr. Levendovszky Tihamér
Dr. Charaf Hassan

BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
AUTOMATIZÁLÁSI ÉS ALKALMAZOTT INFORMATIKAI TANSZÉK

Budapest, 2009.

Ph.D. értekezés tézisei

Bogárdi-Mészöly Ágnes

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

1111 Budapest, Goldmann György tér 3.

e-mail: agi@aut.bme.hu
tel: +36(1)4632486
fax: +36(1)4632871

Tudományos vezetők:

Dr. Levendovszky Tihamér
adjunktus

Dr. Charaf Hassan
docens

1. Előzmények és célkitűzések

A webalapú szoftverrendszerek a felhasználóknak időt és pénzt spórolva segítik az ügyfelek, a szállítók, és az üzleti partnerek közötti együttműködést. E szoftverrendszerek teljesítménye egyike a legfontosabb, de ugyanakkor legbonyolultabb fejlesztési szempontoknak. Közismerten az ilyen rendszerek nagyszámú felhasználó kiszolgálására készülnek, ezért a szolgáltatások effektív elérhetőségét, rendelkezésre állását alacsony válaszidő és adott átbocsátóképesség[1] mellett kell biztosítaniuk.

A terhelés olyan modellezési folyamatként jellemezhető, amely a felhasználói viselkedést reprodukálja [Ferrari 1972]. Az e-kereskedelemben a felhasználók különböző navigációs mintákat, különböző terheléseket használnak. Az egyik megközelítés a Customer Behavior Model Graph-ot [Menascé 2000], a másik a Markov-láncokat [Kijima 1997] alkalmazza. Ezenkívül egyes módszerek a fázis típusú eloszlásokat vagy a kvázi születési-halálozási folyamatokat hasznosítják [Bobbio 2004].

A teljesítménymetrikák sok tényezőtől függnek. E konfigurálható tényezőket számos cikkben vizsgálják, valamint azt is hogyan befolyásolják a webalapú szoftverrendszerek teljesítményét [Sopitkamol 2005]. A teljesítménybefolyásoló tényezők keresésére a statisztikai módszerek és a hipotézisvizsgálatok [Trivedi 1982] hatékonyan alkalmazhatók. A Microsoft .NET [Platt 2003] a webalapú szoftverrendszerek területén egyre nagyobb teret hódít. Egy alkalmazásszerver teljesítményét annak számos beállítása befolyásolhatja [Meier 2004] [Wienholt 2003].

Megfelelően megtervezett teljesítménymodellek és megfelelő kiértékelő algoritmusok segítségével a teljesítménypredikció már a fejlesztési folyamat korábbi fázisaiban is elvégezhető. Ilyen típusú feladatok megoldására különböző módszerek alkalmazhatók. A modellezési technikák egyik csoportja sorbanállási hálózatokon vagy azok kiterjesztésein alapul [Jain 1991] [Kleinrock 1975]. A sorbanállási modellt analitikusan vagy szimulációval megoldva a teljesítménymetrikák előrejelezhetők. A technikák másik csoportja a Petri-hálókat és általánosított sztochasztikus kiterjesztéseiket alkalmazza [Bernardi 2002], melyek a blokkolási és szinkronizációs jelenségeket hatékonyabban tudják modellezni, mint a sorbanállási hálózatok. A harmadik megközelítés pedig a folyamat algebra-sztochasztikus kiterjesztéseit alkalmazza, mint pl. a TIPP (Time Processes and Performability Evaluation) [Herzog 2000], az EMPA (Extended Markovian Process Algebra) [Bernardo 1998] és a PEPA (Performance Evaluation Process Algebra) [Gilmore 1994].

A webalapú szoftverrendszerek a kliensek kéréseinek teljesítése során több erőforráshoz férnek hozzá, tipikusan több kérés érkezik egyidőben, így versengés folyik az erőforrásokért. Ezen helyzet modellezésekor a legtöbb figyelem a sorbanállásimodell-alapú megközelítésekre irányul. A sorbanállási hálózatok széles körben elterjedt eszközök webalapú szoftverrendszerek modellezésére [Menascé 2001] [Urgaonkar 2005].

Kutatásom célja ennek megfelelően webalapú szoftverrendszerek hatékonyságának előrejelzésére alkalmas teljesítménymodellek felállítása és kiértékelési módszerek kidolgozása. Mindezek alapján a következő célokat tűztem ki:

- *A válaszidő és az átbocsátóképesség teljesítménymetrikák tekintetében domináns teljesítménybefolyásoló tényezők azonosítása és vizsgálata.*
- *A válaszidő és az átbocsátóképesség teljesítménymetrikák tekintetében domináns teljesítménybefolyásoló tényezők alkalmazására épülő kiértékelési és predikciós technikák kifejlesztése és vizsgálata.*

[1] throughput: átbocsátóképesség, átvitel

- *Webalapú szoftverrendszerek modellezésére alkalmas többrétegű sorbanállásihálózat-modellek felállítása és verifikálása.*
- *Az ismertetett módszerek alkalmazása valós rendszereknél, illetve a módszerek gyakorlati alkalmazhatóságának igazolása.*

2. Módszertani összefoglalás

A kutatás kiindulópontját a már létező módszerek, modellek, algoritmusok jelentették. Az irodalomkutatást követően a megismert legfontosabb modelleket, algoritmusokat gyűjtöttem össze. Ez segített abban, hogy a probléma megoldásának különböző megközelítéseit jobban megértsem, valamint további új módszerek kidolgozásának alapját képezhetik.

A fenti követelmények határozták meg a kutatásom irányát és az egyes megoldandó részfeladatokat. Az elméleti eredményekkel foglalkozó kutatás módszereit minden esetben az aktuálisan kitűzött feladathoz tartozó módszerek, a már létező részmegoldások, illetve a létező megoldások hiányosságai határozták meg.

Kutatómunkám eredményei, amelyek lehetővé teszik a jelenlegi teljesítménymodellek továbbfejlesztését, három fő részre oszthatók.

- *Teljesítménypredikció és teljesítménybefolyásoló tényezők azonosítása.*
- *Szálkészlet modellezése.*
- *Várakozásisor-hosszak modellezése.*

Megvizsgáltam a jelenlegi teljesítménymodelleket és kiértékelő algoritmusokat webalapú szoftverrendszerek teljesítménymetrikáinak prediktálása céljából, valamint összehasonlítási alapként az általam adott modellek és algoritmusok számára. Új teljesítménybefolyásoló tényezőket kerestem és vizsgáltam. Az identifikált teljesítménybefolyásoló tényezőket modelleztem a teljesítménymodellek továbbfejlesztése céljából.

A kutatási megközelítés egészére érvényes, hogy az elvárt eredmények mindegyike a gyakorlatban is használható megoldást, eredményt célzott meg. A kutatás elméleti eredményei valós rendszereken, és egy általam kifejlesztett webalapú szoftverrendszer adott feladathoz kapcsolódó moduljaiban kerültek megvalósításra.

3. Új tudományos eredmények

Kutatómunkám eredményeit három tézisben foglaltam össze. Az első tézisem többrétegű ASP.NET webalapú szoftverrendszerek modellezésére alkalmas sorbanállásihálózat-modelleket és kiértékelő algoritmusokat ad, amelyekről beláttam, hogy ASP.NET környezetben teljesítménypredikcióra alkalmasak. Tézisemben teljesítménymérésekkel igazolom a modellek és algoritmusok érvényességét, a predikciók helyességét. Az első tézisem a válaszidő és az átbocsátóképesség tekintetében domináns további teljesítménybefolyásoló tényezők keresésével és vizsgálatával is foglalkozik. A jelenlegi teljesítménymodellek továbbfejlesztése céljából az identifikált teljesítménybefolyásoló tényezőket is modellezni kell.

A második és a harmadik tézisem a válaszidő és az átbocsátóképesség teljesítménymetrikák tekintetében domináns tényezők alkalmazására épülő kiértékelési és predikciós technikák kifejlesztésével és elemzésével foglalkozik. A második tézis a szálkészlet hatásainak modellezésére ad algoritmust, a harmadik tézis a várakozásisor-hosszak modellezésével foglalkozik. A disszertáció az algoritmusok komplexitását, valamint a modellek és algoritmusok által adott válaszidő és átbocsátóképesség sorozatok határértékét is vizsgálja. Megmutattam, hogy az általam adott modellek és algoritmusok ASP.NET környezetben teljesítménypredikcióra alkalmasak. Teljesítménymérésekkel igazoltam az általam adott modellek és algoritmusok érvényességét, a predikciók helyességét.

A tézisfüzet további felépítése a következő. Elsőként az adott tézishez tartozó fő eredményeket ismertetem, majd a disszertáció megfelelő fejezetére való hivatkozás után a kapcsolódó publikációk listája szerepel, amit az adott tézis részletes leírása követ.

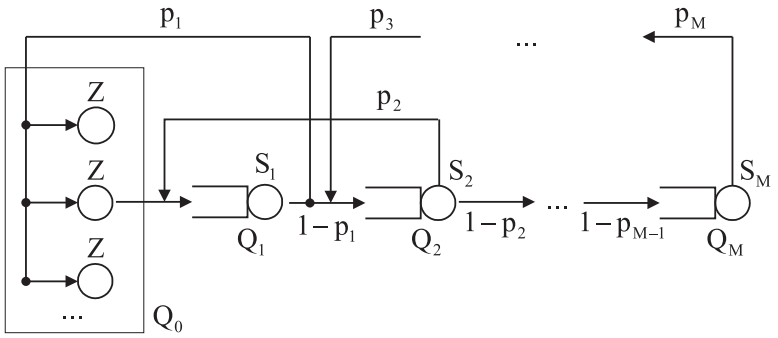
I. Tézis
Teljesítménypredikció és teljesítménybefolyásoló tényezők azonosítása

Sorbanállásihálózat-modellt és kiértékelő algoritmusokat adtam többrétegű ASP.NET webalapú szoftverrendszerek modellezésére. Beláttam, hogy ASP.NET környezetben teljesítménypredikcióra alkalmasak. Teljesítménymérésekkel igazoltam érvényességüket, a predikciók helyességét. Megvizsgáltam komplexitásukat, valamint az általuk adott válaszidő és átbocsátóképesség sorozatok határértékét, melyek összehasonlítási alapként szolgálhatnak az általam adott modellek és algoritmusok számára.

Teljesítménybefolyásoló tényezők azonosítására és vizsgálatára alkalmas statisztikai módszereket javasoltam. Igazoltam, hogy a szálkészlet és a várakozásisor-hosszakat korlátozó paraméterek nagymértékben befolyásolják a teljesítményt, más szóval teljesítménybefolyásoló tényezők. Megmutattam, hogy a válaszidő teljesítménymetrika normális eloszlást közelít. Meghatároztam az eloszlás paramétereit maximum likelihood becsléssel, szukcesszív approximációval.

Az I. tézis a disszertáció 4. fejezetében található. Az I. tézishez kapcsolódó publikációk: [1] [2] [3] [7] [8] [9] [10] [12] [13] [15] [16] [18] [19] [20] [21] [34].

3.1. Definíció. *Az 1. ábrán látható alap sorbanállási modell (base queueing model) többrétegű információs rendszerek modellezésére definiált [Urgaonkar 2005], ahol a* $Q_1, ..., Q_M$ *várakozási sorok az egyes rétegeknek felelnek meg. Egy kérés a végrehajtása során az egyes várakozási sorokat akár többször is meglátogathatja, így minden várakozási sorból az azt megelőző és követő várakozási sorba is van átmenet. Nevezetesen egy kérés a* Q_m*-ből* p_m *valószínűséggel* Q_{m-1}*-be tér vissza,* $1-p_m$ *valószínűséggel pedig* Q_{m+1} *felé továbbítódik. Két kivétel van: az utolsó várakozási sor* Q_M*, ahol minden kérés az előző várakozási sorba tér vissza* ($p_M=1$) *és az első várakozási sor* Q_1*, ahol az előző várakozási sorba való átmenet a kérés teljesítését jelenti.* S_m *jelöli a kérés kiszolgálási idejét a* Q_m *várakozási sornál* ($1 \leq m \leq M$).



1. ábra. Többrétegű információs rendszer modellezése sorbanállási hálózattal

Az internetes terhelések általában munkamenet-alapúak. A modell a munkamenet-alapú terhelést Q_0 végtelen kiszolgálós sorbanállási rendszerként kezeli, amely kérésekkel látja el a várakozási sorok hálózatát és alakítja ki a zárt sorbanállási rendszert. Minden aktív munkamenet egy szervert foglal el Q_0-ban. A Q_0-ban töltött idő a gondolkodási időnek (Z) felel meg.

Egy webalkalmazás bejövő kérései osztályokba sorolhatók. Az alap sorbanállási modell egyik kiterjesztése [Urgaonkar 2005] több kérésosztály kezelésére is alkalmas.

3.2. Definíció. *A középérték analízis algoritmus (Mean-Value Analysis, MVA) [Reiser 1980] az alap sorbanállási modellhez a 3.1. algoritmussal definiált, az alkalmazott jelöléseket az 1. táblázat tartalmazza.*

Algorithm 3.1 Az MVA algoritmus pszeudokódja

1: **for all** $m = 1$ to M **do**
2: $L_m = 0$
3: **for all** $n = 1$ to N **do**
4: **for all** $m = 1$ to M **do**
5: $R_m = V_m \cdot S_m \cdot (1 + L_m)$
6: $R = \sum_{m=1}^{M} R_m$
7: $\tau = n/(Z + R)$
8: **for all** $m = 1$ to M **do**
9: $L_m = \tau \cdot R_m$

Több kérésosztály esetén az MVA idő- és helykomplexitása arányos a lehetséges populációk számával, így már relatívan kevés osztály és felhasználószám esetén is nagy a komplexitás. Ezekben az esetekben MVA helyett megéri közelítő MVA-t vagy kétoldali határszámítást (balanced job bounds) alkalmazni.

Az alkalmazott közelítő MVA algoritmus [Sinclair 2005] pszeudokódja a következő (a jelöléseket lásd az 1. táblázatban).

Algorithm 3.2 Az alkalmazott közelítő MVA algoritmus pszeudokódja

1: **for all** m,c **do**
2: $L_{m,c} = \frac{N_c}{M}$
3: **repeat**
4: $E_{m,c} = \frac{N_c - 1}{N_c} L_{m,c} + \sum_{\substack{i=1 \\ i \neq c}}^{C} L_{m,i}$
5: Compute $R_{m,c}$ and τ_c using $E_{m,c}$
6: Compute $newL_{m,c}$ using τ_c and $R_{m,c}$
7: **until** $\frac{newL_{m,c} - oldL_{m,c}}{oldL_{m,c}} < \delta$

A kétoldali határszámítás azon alapul, hogy egy kiegyensúlyozott rendszer teljesítménye jobb, mint egy nem kiegyensúlyozotté [Zahorjan 1982]. Kiegyensúlyozott rendszernek nevezzük azt a rendszert, amelyben nincs szűk keresztmetszet, azaz minden várakozási sorban azonos a kiszolgálási idő. Az alsó és felső határok nagyon közel esnek egymáshoz, a valódi teljesítményt is jól közelítik.

3.3. Definíció. *A balanced job bounds a 3.1. és a 3.2. egyenletekkel definiált, az alkalmazott jelöléseket az 1. táblázat tartalmazza.*

$$\begin{gathered} \max\left\{ND_{\max} - Z, D + (N-1)D_{avg}\frac{D}{D+Z}\right\} \leq \\ \leq R \leq D + (N-1)D_{\max}\frac{(N-1)D}{(N-1)D+Z} \end{gathered} \tag{3.1}$$

$$\frac{N}{Z + D + (N-1)D_{\max}\frac{(N-1)D}{(N-1)D+Z}} \leq \tau \leq \min\left\{\frac{1}{D_{\max}}, \frac{N}{Z + D + (N-1)D_{avg}\frac{D}{D+Z}}\right\} \tag{3.2}$$

1. táblázat. Kiértékelő algoritmusok jelölései

Jelölés	Jelentés	Jelölés	Jelentés
C	kérésosztályok száma	N	felhasználók száma
D	teljes kiszolgálási igény	N_c	c osztályú kérések száma
D_{avg}	átlagos kiszolgálási igény	R	válaszidő
D_m	kiszolgálási igény Q_m-nél	R_m	válaszidő Q_m-nél
D_{max}	maximális kiszolgálási igény	$R_{m,c}$	c osztályú kérések válaszideje Q_m-nél
δ	maximális megengedhető hiba	S_m	kiszolgálási idő Q_m-nél
$E_{m,c}$	Q_m elvárt hossza c osztályú kéréseknél	τ	átbocsátóképesség
L_m	Q_m hossza	τ_c	c osztályú kérések átbocsátóképessége
$L_{m,c}$	Q_m hossza c osztályú kéréseknél	V_m	látogatási szám Q_m-nél
M	rétegek száma	Z	gondolkodási idő

A webalapú szoftverrendszereket érdemes rétegekre bontani és az 1. ábrán látható módon modellezni, mert az egyes rétegek független és additív kiszolgálási ideje nagymértékben eltérhet egymástól.

I.1. Altézis

Megmutattam, hogy az alap sorbanállási modell (3.1. definíció) és az MVA (3.2. definíció), a közelítő MVA (3.2. algoritmus), a balanced job bounds (3.3. definíció) algoritmusok alkalmasak többrétegű ASP.NET webalapú szoftverrendszerek modellezésére. Beláttam, hogy ASP.NET környezetben teljesítménypredikcióra alkalmasak: a válaszidő, az átbocsátóképesség, és a rétegkihasználtság teljesítménymetrikák prediktálhatók. Teljesítménymérésekkel igazoltam érvényességüket, a predikciók helyességét.

Megvizsgáltam az MVA algoritmus komplexitását, valamint az alap sorbanállási modell és az MVA algoritmus által adott válaszidő és átbocsátóképesség sorozatok határértékét, melyek összehasonlítási alapként szolgálhatnak az általam adott modellek és algoritmusok számára.

A 3.1. rekurzív algoritmus 5. 6. 7. és 9. lépéseit egymásba helyettesítve a válaszidő sorozata a következő egyenlettel írható le:

$$R(n) = \sum_{m=1}^{M} D_m + \frac{(n-1)\sum_{m=1}^{M} D_m R_m(n-1)}{Z + R(n-1)}, \tag{3.3}$$

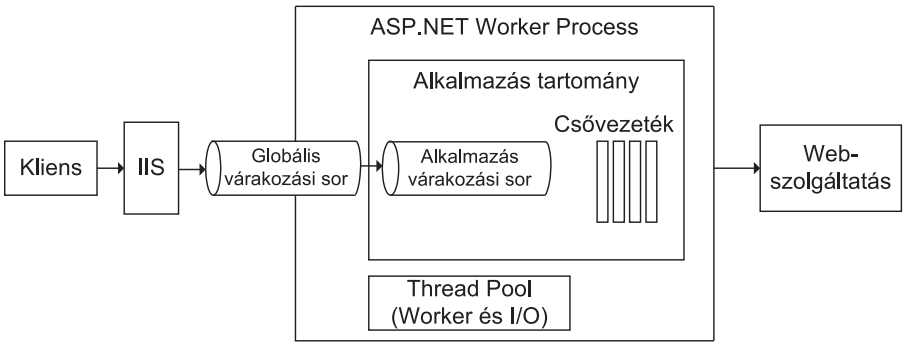
az átbocsátóképesség sorozata pedig a következő:

$$\tau(n) = \frac{n}{Z + (D_1 + D_2 + ... + D_M) + \sum_{i=2}^{n} \left\{ (D_1^i + D_2^i + ... + D_M^i) \prod_{j=1}^{i-1} \tau(n-j) \right\}}. \tag{3.4}$$

I.2. Altézis

Beláttam, hogy az MVA algoritmussal (3.2. definíció) az alap sorbanállási modell (3.1. definíció) kiértékelhető. Igazoltam, hogy az MVA algoritmus komplexitása $\Theta(N \cdot M)$, ahol N a felhasználók, M a rétegek száma. Megmutattam, hogy az alap sorbanállási modell és az MVA algoritmus által adott válaszidő sorozat (3.3. egyenlet) végtelenhez konvergál (ha $n \to \infty$, ahol n a felhasználók száma). Bebizonyítottam, hogy az alap sorbanállási modell és az MVA algoritmus által adott átbocsátóképesség sorozat (3.4. egyenlet) határértéke $1/D_{\max}$ (ha $n \to \infty$, ahol n a felhasználók száma), ahol $D_{\max}$ a maximális kiszolgálási igény.

A jelenlegi teljesítménymodellek és kiértékelő algoritmusok elemzése után új teljesítménybefolyásoló tényezőket kerestem és vizsgáltam a teljesítménymodellek továbbfejlesztése céljából. Az értekezés a vizsgálatok módszertanát illetően statisztikai módszereket javasol.



2. ábra. Az ASP.NET környezet architektúrája

Szálkészlet esetében az alkalmazás a beérkező kéréseket várakozási sorba helyezi [Carmona 2002] (2. ábra). A szálak ebből a várakozási sorból veszik ki a kéréseket és dolgozzák fel azokat. Amikor egy szál felszabadul, a következő kérést veszi ki és hajtja végre. A várakozási sor méretét célszerű limitálni annak érdekében, hogy a beérkező kérések ne telíthessék meg a szerver- valamint az alkalmazás-várakozásisorhoz rendelt memóriát.

Az IIS-ből a HTTP-kérések a megnevezett csővezetékbe kerülnek, amely egy globális várakozási sor az IIS és az ASP.NET között, ahol a kérések natív kódból a felügyelt szálkészlethez továbbítódnak. Ezt a globális várakozási sort az ASP.NET által futtatott folyamat kezeli, és a *requestQueueLimit* tulajdonság által konfigurálható. Amikor a „Requests Current" számláló – amely a sorba állított, végrehajtás alatt álló, és a kliensre várakozó kéréseket egyaránt tartalmazza – eléri ezt a küszöbértéket, a további kérések elutasításra kerülnek [Marquardt 2003].

A globális várakozási sorból a kérések egy alkalmazás-várakozásisorba kerülnek, amely az *appRequestQueueLimit* tulajdonság által konfigurálható. Minden alkalmazástartományhoz külön alkalmazás-várakozásisor tartozik, amelyet a worker és I/O szálak elérhetősége is befolyásol. A kérések száma akkor nő ebben a várakozási sorban, ha az elérhető worker és I/O szálak száma a *minFreeThreads* tulajdonságban beállított érték alá esik. Ha a kérések száma eléri a küszöbértéket, akkor a szerver visszautasítja a kéréseket.

A *maxWorkerThreads* tulajdonság a worker szálak, a *maxIOThreads* az I/O szálak maximális számát határozza meg a .NET-szálkészletben (thread pool). A *minFreeThreads* tulajdonság a konkurens kérések számát korlátozza, oly módon, hogy minden bejövő kérés várakozási sorba kerül, ha az elérhető szálak száma a szálkészletben ezen érték alá esik. A *minLocalRequestFreeThreads* hasonló a *minFreeThreads*-hez, csak lokális kérésekre vonatkozik. Ez utóbbi két tulajdonság a holtpontok megelőzése miatt fontos.

I.3. Altézis

Beláttam, hogy a khi-négyzet függetlenségvizsgálat teljesítménybefolyásoló tényező azonosítására alkalmas. Igazoltam, hogy a szálkészlet szálait (maxWorkerThreads, maxIOThreads, minFreeThreads, és minLocalRequestFreeThreads), valamint a várakozásisorhosszakat (requestQueueLimit és appRequestQueueLimit) korlátozó paraméterek teljesítménybefolyásoló tényezők. Statisztikai módszerekkel megmutattam, hogy a válaszidő teljesítménymetrika normális eloszlást közelít, ha a valószínűségi változók a szálkészlet paramétereknek felelnek meg (név szerint maxWorkerThreads, maxIOThreads, minFreeThreads és minLocalRequestFreeThreads). Meghatároztam az eloszlás paramétereit maximum likelihood becsléssel, szukcesszív approximációval.

A webalapú szoftverrendszerek teljesítménymodelljeinek továbbfejlesztése érdekében az általam azonosított teljesítménybefolyásoló tényezők (szálkészlet és várakozásisorhossz) modellezése elengedhetetlen.

II. Tézis
Szálkészlet modellezése

Algoritmus dolgoztam ki a szálkészlet teljesítménybefolyásoló tényező hatásainak modellezésére. Megvizsgáltam az általam adott algoritmus komplexitását, valamint az algoritmus által prediktált válaszidő és átbocsátóképesség sorozatok határértékét. Beláttam, hogy a javasolt algoritmus ASP.NET környezetben webalapú rendszerek teljesítménypredikciójára alkalmas: a válaszidő és az átbocsátóképesség teljesítménymetrikák prediktálhatók. Teljesítménymérésekkel igazoltam az általam adott algoritmus érvényességét, az algoritmussal végzett predikciók helyességét. Megmutattam, hogy az általam adott algoritmus átlagos abszolút hibája kisebb az eredeti algoritmus átlagos abszolút hibájánál. Megvizsgáltam a javasolt és az eredeti algoritmusok hiba hisztogramjait is.

Az II. tézis a disszertáció 5. fejezetében található. Az II. tézishez kapcsolódó publikációk: [4] [5] [6] [22] [23] [24] [25] [26] [27].

A szálkészlet viselkedését figyelembe véve az MVA kiértékelő algoritmus (3.2. definíció) hatékonyan továbbfejleszthető.

3.4. Definíció. *A szálkészlettel kibővített MVA (extended MVA with thread pool, MVA-TP) a 3.3. algoritmus által definiált, ahol a CPU index a CPU réteg indexe, az I/O index pedig egy a* $1 \leq m \leq M$ *intervallumból vett I/O réteg indexnek felel meg.*

Algorithm 3.3 Az MVA-TP algoritmus pszeudokódja

for all $m = 1$ to M **do**
  $L_m = 0$
for all $n = 1$ to N **do**
  for all $m = 1$ to M **do**
    $R_m = V_m \cdot S_m \cdot (1 + L_m)$
  $R = \sum_{m=1}^{M} R_m$
  $\tau = n/(Z + R)$
  for all $m = 1$ to M **do**
    if *m is an I/O tier* **then**
      $L_{I/O} = \tau \cdot R_{I/O} - \frac{S_{CPU}}{S_{I/O}} \cdot L_{CPU}$
      if $L_{I/O} < 0$ **then**
        $L_{I/O} = 0$
  for all $m = 1$ to M **do**
    if *if m is a CPU tier* **then**
      $L_m = \tau \cdot R_m$

II.1. Altézis

Algoritmust (3.4. definíció) dolgoztam ki a szálkészlet hatásainak modellezésére. Megmutattam, hogy az általam adott MVA-TP algoritmussal az alap sorbanállási modell (3.1. definíció) kiértékelhető. Igazoltam, hogy az algoritmus komplexitása $\Theta(N \cdot M)$, ahol N a felhasználók, M a rétegek száma.

Ez a kibővítés nem növeli a kiértékelő algoritmus komplexitását, mivel az megegyezik az eredeti algoritmus komplexitásával (lásd I.2. Altézis).

II.2. Altézis

Bebizonyítottam, hogy az általam adott MVA-TP algoritmus (3.4. definíció) által prediktált átbocsátóképesség sorozat határértéke $1/D_{\max}$ (ha $n \to \infty$, ahol n a felhasználók száma), ahol $D_{\max}$ a maximális kiszolgálási igény. Megmutattam, hogy az MVA-TP algoritmus által prediktált válaszidő sorozat végtelenhez konvergál (ha $n \to \infty$, ahol n a felhasználók száma). Beláttam, hogy a javasolt MVA-TP és az eredeti MVA (3.2. definíció) különbsége az átbocsátóképesség és a válaszidő esetén is nullához konvergál (ha $n \to \infty$, ahol n a felhasználók száma).

Az általam adott MVA-TP algoritmus által prediktált válaszidő és az átbocsátóképesség sorozatok határértéke megegyezik az eredeti algoritmus által számolt teljesítménymetrikák határértékeivel (lásd I.2. Altézis).

$\lim_{n\to\infty} a_n = A$, a határérték definíciója a következő: $\forall \varepsilon > 0$ $(\varepsilon \in R)$ $\exists N(\varepsilon) \in N$ hogy $|a_n - A| < \varepsilon$ ha $n > N(\varepsilon)$. A definíció alapján minden tetszőleges ε-hoz létezik egy közös küszöbindex: $\max(N_1(\varepsilon), N_2(\varepsilon))$, ahol $N_1(\varepsilon)$ az általam adott és az eredeti algoritmusok által prediktált válaszidő különbség sorozathoz, valamint $N_2(\varepsilon)$ az átbocsátóképesség különbség sorozathoz tartozik. Ha a felhasználók száma nagyobb, mint a küszöbindex, az általam adott MVA-TP és az eredeti MVA is használható. Ha a felhasználók száma kisebb, mint a küszöbindex, akkor az általam adott MVA-TP algoritmust kell alkalmazni.

A teljesítménypredikciók helyességének verifikálására két módszert alkalmaztam: az átlagos abszolút hibafüggvényt és a hiba hisztogramot.

3.5. Definíció. *Az átlagos abszolút hibafüggvény a 3.5. egyenlettel definiált, ahol az alg index az algoritmus által prediktált, az obs index pedig a megfigyelt értékeket jelenti, valamint R a válaszidő, és N a mérések száma.*

$$error_{alg-obs} = (\sum_{i=1}^{N} |R_{alg_i} - R_{obs_i}|)/N. \tag{3.5}$$

II.3. Altézis

Megmutattam, hogy az általam adott MVA-TP algoritmus (3.4. definíció) ASP.NET környezetben webalapú szoftverrendszerek teljesítménypredikciójára alkalmas, a javasolt algoritmus az eredeti algoritmusnál (3.2. definíció) pontosabban prediktálja a teljesítménymetrikákat. Teljesítménymérésekkel igazoltam a javasolt MVA-TP érvényességét, az algoritmussal végzett predikciók helyességét. Megmutattam, hogy az általam adott MVA-TP algoritmus átlagos abszolút hibája (3.5. definíció) kisebb az eredeti MVA algoritmus átlagos abszolút hibájánál. Megadtam a javasolt MVA-TP és az eredeti MVA algoritmusok hiba hisztogramjait is. Az átlagos abszolút hibafüggvény és a hiba hisztogram eredményei is azt mutatják, hogy az általam adott algoritmus az eredeti algoritmusnál pontosabban prediktálja a teljesítménymetrikákat.

III. Tézis
Várakozásisor-hosszak modellezése

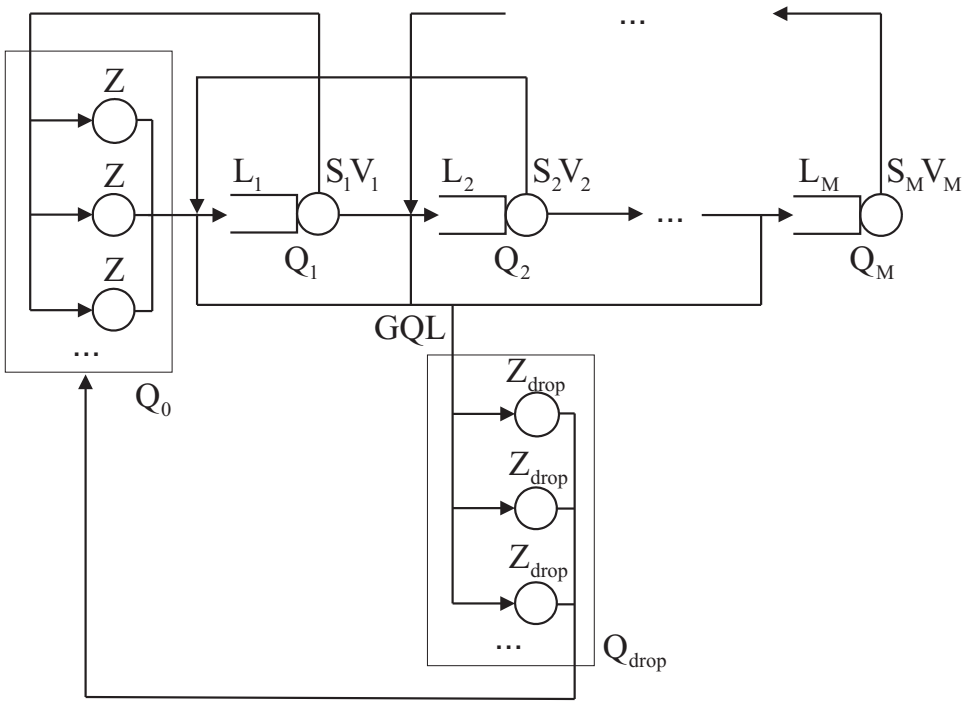
Sorbanállási modelleket adtam és kiértékelő algoritmusokat dolgoztam ki a várakozásisor-hossz teljesítménybefolyásoló tényező modellezésére. Megvizsgáltam az általam adott algoritmusok komplexitását, valamint a javasolt modellek és algoritmusok által prediktált válaszidő és átbocsátóképesség sorozatok határértékét. Megmutattam, hogy a javasolt modellek és algoritmusok ASP.NET környezetben webalapú rendszerek teljesítménypredikciójára alkalmasak: a válaszidő és az átbocsátóképesség teljesítménymetrikák előrejelezhetők. Teljesítménymérésekkel bizonyítottam az általam adott modellek és algoritmusok érvényességét, a predikciók helyességét. Beláttam, hogy az általam adott modellek és algoritmusok átlagos abszolút hibája kisebb az eredeti modell és algoritmus átlagos abszolút hibájánál. Megvizsgáltam a javasolt és az eredeti modellek és algoritmusok hiba hisztogramjait is.

Az III. tézis a disszertáció 6. fejezetében található. Az III. tézishez kapcsolódó publikációk: [28] [29] [30] [31] [32].

A várakozásisor-hosszakat figyelembe véve az alap sorbanállási modell (3.1. definíció) és az MVA kiértékelő algoritmus (3.2. definíció) hatékonyan kibővíthető.

3.6. Definíció. *A globális várakozásisor-hosszal kibővített sorbanállási modell (extended queueing model with global queue limit, QM-GQL) a 3. ábra szerint definiált, ahol Q_{drop} végtelen kiszolgálós sorbanállási rendszer, Z_{drop} a Q_{drop}-ban eltöltött idő, GQL a globális várakozásisor-hossz maximuma, amely ASP.NET környezetben a requestQueueLimit paraméternek felel meg. Ha GQL kisebb, mint a sorbanálló kérések számának összege ($\sum_{m=1}^{M} L_m$), a következő kérések a Q_{drop}-ba kerülnek. Q_{drop}-ból a kérések pedig Q_0 továbbítódnak, vagyis újra kibocsátásra kerülnek.*

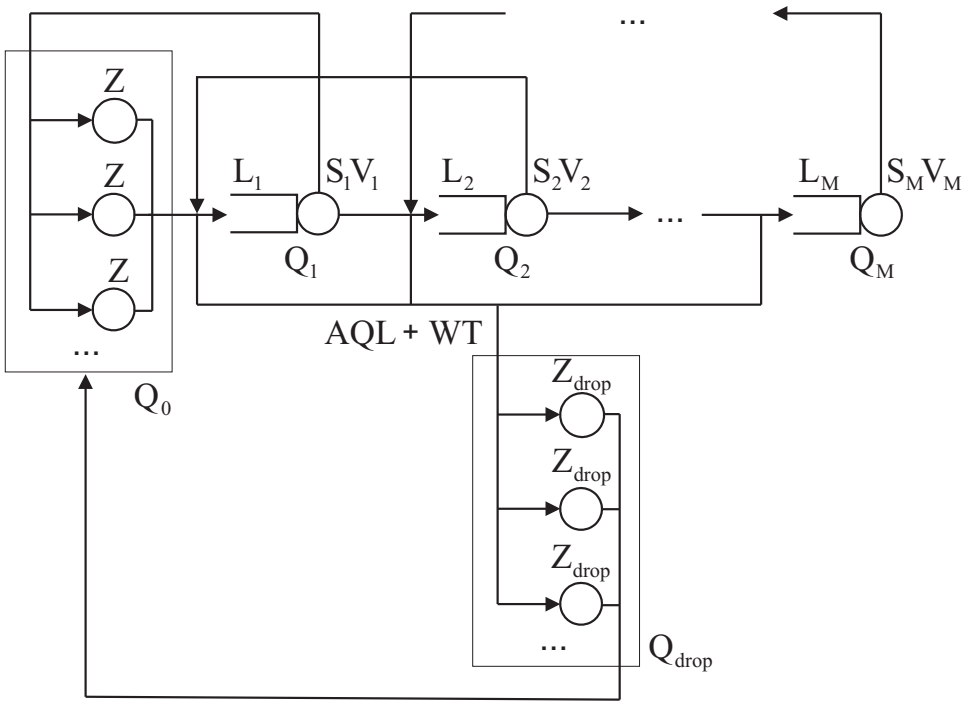
3.7. Definíció. *A globális várakozásisor-hosszal kibővített MVA (extended MVA with global queue limit, MVA-GQL) a 3.4. algoritmus által definiált, ahol Z_{drop} a Q_{drop}-ban eltöltött idő, GQL a globális várakozásisor-hossz maximuma, amely ASP.NET környezetben a requestQueueLimit paraméternek felel meg.*



3. ábra. A QM-GQL modell

Algorithm 3.4 Az MVA-GQL algoritmus pszeudokódja

for all $m = 1$ to M **do**
  $L_m = 0$
$nql = 1$
for all $n = 1$ to N **do**
  for all $m = 1$ to M **do**
    $R_m = V_m \cdot S_m \cdot (1 + L_m)$
  $R = \sum_{m=1}^{M} R_m$
  $\tau = nql/(Z + Z_{drop} + R)$
  for all $m = 1$ to M **do**
    $L_m = \tau \cdot R_m$
  if $\sum_{m=1}^{M} L_m > GQL$ **then**
    for all $m = 1$ to M **do**
      $L_m = oldL_m$
  else
    $nql = nql + 1$
  for all $m = 1$ to M **do**
    $oldL_m = L_m$



4. ábra. A QM-AQL modell

Algorithm 3.5 Az MVA-AQL algoritmus pszeudokódja

for all $m = 1$ to M **do**
  $L_m = 0$
$nql = 1$
for all $n = 1$ to N **do**
  for all $m = 1$ to M **do**
    $R_m = V_m \cdot S_m \cdot (1 + L_m)$
  $R = \sum_{m=1}^{M} R_m$
  $\tau = nql/(Z + Z_{drop} + R)$
  for all $m = 1$ to M **do**
    $L_m = \tau \cdot R_m$
  if $\sum_{m=1}^{M} L_m - WT > AQL$ **then**
    for all $m = 1$ to M **do**
      $L_m = oldL_m$
  else
    $nql = nql + 1$
  for all $m = 1$ to M **do**
    $oldL_m = L_m$

3.8. Definíció. *Az alkalmazás-várakozásisorhosszal kibővített sorbanállási modell (extended queueing model with application queue limit, QM-AQL) a 4. ábra szerint definiált, ahol Q_{drop} végtelen kiszolgálós sorbanállási rendszer, Z_{drop} a Q_{drop}-ban eltöltött idő, AQL az alkalmazás-várakozásisorhossz maximuma, amely ASP.NET környezetben az appRequestQueueLimit paraméternek felel meg, valamint WT a dolgozó szálakat jelenti, amely ASP.NET környezetben egyenlő a $maxWorkerThreads + maxIOThreads - -minFreeThreads$ értékkel. Ha $AQL+WT$ kisebb, mint a sorbanálló kérések számának összege ($\sum_{m=1}^{M} L_m$), a következő kérések a Q_{drop}-ba kerülnek. Q_{drop}-ból a kérések pedig Q_0 továbbítódnak, vagyis újra kibocsátásra kerülnek.*

3.9. Definíció. *Az alkalmazás-várakozásisorhosszal kibővített MVA (extended MVA with application queue limit, MVA-AQL) a 3.5. algoritmus által definiált, ahol Z_{drop} a Q_{drop}-ban eltöltött idő, AQL az alkalmazás-várakozásisorhossz maximuma, amely ASP.NET környezetben az appRequestQueueLimit paraméternek felel meg, valamint WT a dolgozó szálakat jelenti, amely ASP.NET környezetben egyenlő a $maxWorkerThreads+maxIOThreads- -minFreeThreads$ értékkel.*

III.1. Altézis

Sorbanállási modelleket (3.6 és 3.8. definíció) adtam és kiértékelő algoritmusokat (3.7 és 3.9. definíció) dolgoztam ki a globális és alkalmazás-várakozásisorhosszak modellezésére. Megmutattam, hogy az MVA-GQL közelítő algoritmussal a QM-GQL modell, az MVA-AQL közelítő algoritmussal pedig a QM-AQL modell kiértékelhető. Igazoltam, hogy az általam adott algoritmusok komplexitása $\Theta(N \cdot M)$, ahol N a felhasználók, M a rétegek száma.

Ezek a kibővítések nem növelik a kiértékelő algoritmus komplexitását, mivel azok megegyeznek az eredeti algoritmus komplexitásával (lásd I.2. Altézis).

III.2. Altézis

Megmutattam, hogy az általam adott modellek és algoritmusok (3.6. és 3.7. definíció, valamint 3.8. és 3.9. definíció) által prediktált válaszidő sorozat határértéke RQL (ha $n \to \infty$, n a felhasználók száma), ahol RQL konstans értéket a 3.6. egyenlet adja, QL pedig az az index, amikor $\sum_{m=1}^{M} L_m$ elsőként nagyobb mint GQL vagy $AQL+WT$. Bebizonyítottam, hogy a javasolt modellek és algoritmusok által prediktált átbocsátóképesség sorozat $\min\{1/D_{\max}, \tau QL\}$-hoz konvergál (ha $n \to \infty$, n a felhasználók száma), ahol $D_{\max}$ a maximális kiszolgálási igény, továbbá τQL konstans értéket a 3.7. egyenlet adja, QL pedig az az index, amikor $\sum_{m=1}^{M} L_m$ elsőként nagyobb mint GQL vagy $AQL+WT$.

$$RQL = \sum_{m=1}^{M} D_m + \frac{QL \sum_{m=1}^{M} D_m R_m(QL)}{Z + Z_{drop} + R(QL)} \tag{3.6}$$

$$\tau QL = \frac{QL}{Z + Z_{drop} + \sum_{m=1}^{M} D_m + \sum_{i=2}^{QL} \left(\sum_{m=1}^{M} D_m^i \prod_{j=1}^{i-1} \tau(QL-j) \right)} \tag{3.7}$$

Amíg valamennyi kérés kiszolgálása sikeres, az általam adott modellek és algoritmusok által prediktált válaszidő és átbocsátóképesség megegyezik az eredeti modell és algoritmus által adott teljesítménymetrikákkal. Amennyiben a sorhossz túllépése miatt a rendszer kéréseket kényszerül elutasítani, a javasolt modellek és algoritmusok által prediktált átbocsátóképesség sorozat egy konstans értékhez konvergál, ez az érték a legtöbb esetben megegyezik az eredeti modell és algoritmus által meghatározott értékkel, néhány esetben pedig különbözik. Továbbá az általam adott modellek és algoritmusok által prediktált válaszidő sorozat határértéke nem végtelen, hanem egy konstans értékhez konvergál.

III.3. Altézis

Megmutattam, hogy az általam adott modellek és algoritmusok (3.6. és 3.7. definíció, valamint 3.8. és 3.9. definíció) ASP.NET környezetben webalapú szoftverrendszerek teljesítménypredikciójára alkalmasak, a javasolt modellek és algoritmusok az eredeti modellnél és algoritmusnál (3.1. és 3.2. definíció) pontosabban prediktálják a teljesítménymetrikákat. Teljesítménymérésekkel igazoltam az általam adott modellek és algoritmusok érvényességét, az algoritmussal végzett predikciók helyességét. Megmutattam, hogy az általam adott modellek és algoritmusok átlagos abszolút hibája (3.5. definíció) kisebb az eredeti modell és algoritmus átlagos abszolút hibájánál. Megadtam a javasolt és az eredeti modellek és algoritmusok hiba hisztogramjait is. Az átlagos abszolút hibafüggvény és a hiba hisztogram eredményei is azt mutatják, hogy a javasolt modellek és algoritmusok az eredeti modellnél és algoritmusnál pontosabban prediktálják a teljesítménymetrikákat.

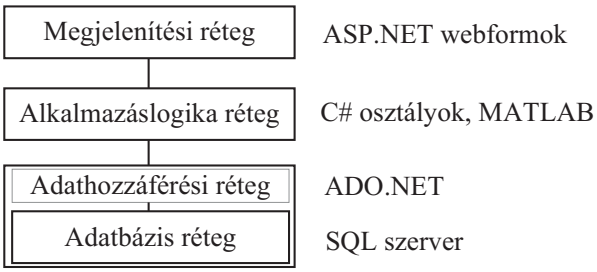
A szálkészlet és a várakozásisor-hossz bővítmények függetlenek és additívak, így együtt és külön-külön is használhatók.

4. Az új tudományos eredmények alkalmazása

Az új tudományos eredmények alkalmazása a disszertáció 7. fejezetében található. Az alkalmazásokkal kapcsolatos publikációk: [28][29].

Az eredmények – teljesítménypredikció és teljesítménybefolyásoló tényezők azonosítása, szálkészlet modellezése, várakozásisor-hosszak modellezése – együttese webalapú szoftverrendszerek továbbfejlesztett teljesítménymodelljeit képezi. Ezeket a technikákat sikeresen alkalmaztam valós rendszereknél, mint pl. a Microsoft Felsőoktatási Portál [MsPortal]. Továbbá kifejlesztettem egy webalapú szoftverrendszert az általam adott módszerek gyakorlati alkalmazhatóságának igazolására.

A disszertáció eredményeit az alábbi szoftvercsomaggal valósítottam meg, amely a disszertáció eredménycsoportjait moduláris felépítéssel tartalmazza. A disszertáció folyamatábrákat és esettanulmányokat is tartalmaz, melyek lehetővé teszik az alkalmazhatóság bemutatását mind a tudományos világ, mint az ipar számára.



5. ábra. A kifejlesztett webalapú szoftverrendszer architektúrája

Az általam kifejlesztett webalapú szoftverrendszer architektúrája az 5. ábrán látható. A javasolt módszereket MATLAB segítségével valósítottam meg, melyeket .NET környezetben alkalmaztam.

Az alkalmazás füles elrendezéssel három fő részre tagolódik:

- *Teljesítmény mérése.*

  Módszertant adtam a teljesítménymérések elvégzéséhez szükséges folyamat leírására. A teljes folyamat nem automatizálható. A mérési folyamat a megadott lépéseket követve végezhető el. A teljesítménymérési folyamat eredményeit az I. tézisben statisztikai módszerekkel analizáltam, továbbá az I. tézisben a jelenlegi, a II. és a III. tézisben pedig az általam adott modellek és algoritmusok validálására, a teljesítménypredikciók helyességének verifikálására használtam. A modell paramétereinek becsléséhez és a modell kiértékeléséhez csupán egy mérésre vagy becslésre van szükség egyetlen felhasználó szimulálása mellett.

- *Teljesítménybefolyásoló tényezők azonosítása.*

  A teljesítménybefolyásoló tényező azonosításának folyamatát dolgoztam ki. Az I. tézisben javasolt statisztikai módszereket valósítottam meg itt.

- *Teljesítménymetrikák előrejelzése, teljesítménymodellek és kiértékelő algoritmusok validálása, hiba elemzése.*

  A következő funkciókat valósítottam meg. A teljesítménymetrikák prediktálhatók az eredeti és az általam adott – a szálkészlet hatásait és a várakozásisor-hosszakat figyelembe vevő – modellekkel és algoritmusokkal. Az általam adott modellek és algoritmusok teljesítménymérésekkel validálhatók. A javasolt modellek és algoritmusok pontossága hiba analízissel ellenőrizhető. Az I. II. valamint III. tézis eredménycsoportjait valósítottam meg itt.

5. Az értekezés témakörében készült tudományos közlemények

Idegen nyelvű folyóiratcikkek

1 **Bogárdi-Mészöly Á.**, Imre G. és Charaf H., „Investigating Factors Influencing the Response Time in J2EE Web Applications", WSEAS Transactions on Computers, 4(2). szám, 179–183.o., 2005. február, ISSN 1109-2750.

2 **Bogárdi-Mészöly Á.**, Szitás Z., Levendovszky T. és Charaf H., „Investigating Factors Influencing the Response Time in ASP.NET Web Applications", Advances in Informatics, Lecture Notes in Computer Science, 3746. szám, 223–233.o., ISSN 0302-9743, IF: 0.402, 2005. november.

3 **Bogárdi-Mészöly Á.**, Levendovszky T. és Charaf H., „Models for Predicting the Performance of ASP.NET Web Applications", Periodica Polytechnica Electrical Engineering, 51(3-4). szám, pp. 111–118, 2007.

4 **Bogárdi-Mészöly Á.**, Hashimoto T., Levendovszky T. és Charaf H., „Thread Pool-Based Improvement of the Mean-Value Analysis Algorithm", Lecture Notes in Electrical Engineering, 27. szám, 1241–1254.o., ISBN 978-0-387-84813-6, 2009. február.

5 **Bogárdi-Mészöly Á.**, Levendovszky T. és Charaf H., „An Improved Algorithm for Performance Prediction of Multi-Tier Information Systems", Performance Evaluation, Elsevier, átdolgozás alatt.

6 **Bogárdi-Mészöly Á.**, Levendovszky T. és Charaf H., „Improved Performance Model for Web-Based Software Systems", WSEAS Transactions, bírálat alatt.

Magyar nyelvű folyóiratcikkek

7 **Bogárdi-Mészöly Á.**, Imre G. és Levendovszky T., „ASP.NET és Java EE webalkalmazások teljesítménypredikciója sorbanállási modell segítségével", Magyar Távközlés, 2006/3. szám.

Nemzetközi konferencia-kiadványban megjelent idegen nyelvű előadások

8 **Bogárdi-Mészöly Á.** és Charaf H., „Comparison of Portal Building Techniques", microCAD 2004 International Scientific Conference, Miskolc, 2004. március 18-19., 41–46.o.

9 **Bogárdi-Mészöly Á.**, Imre G., Levendovszky T. és Charaf H., „Determining the Distribution of the Response Time in J2EE Web Applications", microCAD 2005 International Scientific Conference, Miskolc, 2005. március 10-11., 33–38.o.

10 **Bogárdi-Mészöly Á.**, Imre G., Levendovszky T. és Charaf H., „Investigating the Response Time in J2EE Web Applications", 5th International Conference of PhD Students, Miskolc, 2005. augusztus 14-20., 19–24.o.

11 Imre G., **Bogárdi-Mészöly Á.** és Charaf H., „Monitoring the Performance of J2EE Web Applications", 5th International Conference of PhD Students, Miskolc, 2005. augusztus 14-20., 13–18.o.

12 **Bogárdi-Mészöly Á.**, Levendovszky T. és Charaf H., „Extending the Performance Models of Web Applications with Queueing Algorithm", 6th International Symposium of Hungarian Researchers on Computational Intelligence, Budapest, 2005. november 18-19., 719–730.o.

13 **Bogárdi-Mészöly Á.**, Szitás Z., Levendovszky T. és Charaf H., „Balanced Job Bounds Calculation for Approximating ASP.NET Performance Factors", 4th Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence (SAMI 2006), Szlovákia, Herl'any, 2006. január 20-21., 152–163.o.

14 Imre G., **Bogárdi-Mészöly Á.** és Charaf H., „Measuring and Modelling the Effect of Application Server Tuning Parameters on Performance", 4th Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence (SAMI 2006), Szlovákia, Herl'any, 2006. január 20-21., 471–482.o.

15 **Bogárdi-Mészöly Á.**, Levendovszky T. és Charaf H., „Using Queueing Model in Predicting the Response Time of ASP.NET Web Applications", IASTED International Conference on Software Engineering, Ausztria, Innsbruck, 2006. február 14-16., ACTA Press, IEEE Catalog Number 07EX1642C, ISBN 1-4244-0865-2, 252–257.o.

16 **Bogárdi-Mészöly Á.**, Imre G., Levendovszky T. és Charaf H., „Handling Multiple Session Classes in ASP.NET Environment", microCAD 2006 International Scientific Conference, Miskolc, 2006. március 16-17., 31–36.o.

17 Imre G., **Bogárdi-Mészöly Á.**, Levendovszky T. és Charaf H., „Performance Modelling of a J2EE Web Application Considering Application Server Tuning Parameters", microCAD 2006 International Scientific Conference, Miskolc, 2006. március 16-17., 115–121.o.

18 **Bogárdi-Mészöly Á.**, Levendovszky T. és Charaf H., „Predicting the Performance of ASP.NET Web-Based Information Systems with Optimized Algorithms", 9th International Conference on Information Systems Implementation and Modelling (ISIM 2006), Csehország, Přerov, 2006. április 24-27., 167–174.o.

19 **Bogárdi-Mészöly Á.**, Levendovszky T. és Charaf H., „Handling Session Classes for Predicting ASP.NET Performance Metrics", 4th International Conference in Central Europe on .NET Technologies, Csehország, Plzen, 2006. május 29. - június 1., ISBN 80-86943-11-9, 1–8.o.

20 **Bogárdi-Mészöly Á.**, „Analytical Model for Multi-tier ASP.NET Web Applications", Automation and Applied Computer Science Workshop (AACS 2006), Budapest, 2006. június 30., 109–120.o.

21 **Bogárdi-Mészöly Á.**, Levendovszky T. és Charaf H., „Performance Factors in ASP.NET Web Applications with Limited Queue Models", 10th IEEE International Conference on Intelligent Engineering Systems (INES 2006), Anglia, London, 2006. június 26-28., IEEE Catalog Number 06EX1430, ISBN 1-4244-9708-8, 253–257.o.

22 **Bogárdi-Mészöly Á.**, „An Enhanced Performance Evaluation Algorithm for Enterprise Architectures", Automation and Applied Computer Science Workshop (AACS 2007), Budapest, 2007. június 29., 81–92.o.

23 **Bogárdi-Mészöly Á.**, Hashimoto T., Levendovszky T. és Charaf H., „Improved Evaluation Algorithm for Performance Prediction with Error Analysis", 11th IEEE International Conference on Intelligent Engineering Systems (INES 2007), Budapest, 2007. június 29. - július 1., IEEE Catalog Number 07EX1751C, ISBN 1-4244-1148-3, 301–306.o.

24 **Bogárdi-Mészöly Á.**, Levendovszky T. és Charaf H., „Extending the Mean-Value Analysis Algorithm According to the Thread Pool Investigation", 5th IEEE International Conference on Industrial Informatics (INDIN 2007), Ausztria, Bécs, 2007. július 23-27., IEEE Catalog Number 07EX1642C, ISBN 1-4244-0865-2, ISSN 1935-4576, 731–736.o.

25 **Bogárdi-Mészöly Á.**, Levendovszky T., Charaf H. és Szeghegyi Á., „Effect Analysis of an Improved Performance Evaluation Algorithm", 6th International Symposium on Applied Machine Intelligence and Informatics (SAMI 2008), Szlovákia, Herl'any, 2008. január 21-22., IEEE Catalog Number CFP0808E-CDR, ISBN 978-1-4244-2106-0, 125–130.o.

26 **Bogárdi-Mészöly Á.**, Levendovszky T. és Szeghegyi Á., „Analyzing the Convergence of an Enhanced Performance Evaluation Algorithm", 12th IEEE International Conference on Intelligent Engineering Systems (INES 2008), USA, Florida, Miami, 2008. február 25-29., IEEE Catalog Number CFP08IES-CDR, ISBN 978-1-4244-2083-4, 185–190.o.

27 **Bogárdi-Mészöly Á.**, Levendovszky T., Charaf H. és Szeghegyi Á., „Convergence and Limit of Mean-Value Analysis Algorithms", 12th WSEAS International Conference on Computers (CSCC 2008), New Aspects of Computers, Görögország, Héraklion, 2008. július 23-25., ISBN 978-960-6766-85-5, ISSN 1790-5109, 601–606.o.

28 **Bogárdi-Mészöly Á.** és Levendovszky T., „Application of Improve Performance Models", 9th International Symposium of Hungarian Researchers on Computational Intelligence and Informatics, Budapest, 2008. november 6-8., 205–216.o.

29 **Bogárdi-Mészöly Á.**, Levendovszky T. és Rövid A., „Improved Performance Models and Algorithms", 6th IEEE International Conference on Computational Cybernetics, Szlovákia (ICCC 2008), Stará Lesná, 2008. november 27-29., IEEE Catalog Number CFP08575-CDR, ISBN 978-1-4244-2875-5, 157–162.o.

30 **Bogárdi-Mészöly Á.**, Levendovszky T. és Rövid A., „A Novel Algorithm to Model the Queue Limit", 7th WSEAS International Conference on Circuits, Systems, Electronics, Control & Signal Processing, Spanyolország, Kanári-szigetek, Puerto De La Cruz, 2008. december 15-17., ISBN 978-960-474-037-6, 81–86.o.

31 **Bogárdi-Mészöly Á.**, Levendovszky T. és Szeghegyi Á., „Improved Performance Models of Web-Based Software Systems", 13th IEEE International Conference on Intelligent Engineering Systems, Barbados, 2009. április 16-18., IEEE Catalog Number CFP09IES-CDR, ISBN 978-1-4244-4113-6, 33–38.o.

32 **Bogárdi-Mészöly Á.**, Levendovszky T. és Szeghegyi Á., „Performance Prediction with Algorithm Modeling the Queue Limit", 13th World Multi-Conference on Systemics, Cybernetics and Informatics), USA, Florida, Orlando, 2009. július 10-13., megjelenés alatt.

Technical Report

33 Imre G. és **Bogárdi-Mészöly Á.**, „J2EE Enterprise rendszerek vizsgálata", Kutatási-fejlesztési jelentés, T-Mobile Magyarország Távközlési Rt., 2004. november.

34 **Bogárdi-Mészöly Á.**, „ASP.NET webalkalmazások teljesítménypredikciója", Kutatási jelentés, evosoft Hungary Kft., 2006. október.

6. Idegen hivatkozások

[15] publikációra hivatkozik:

Gorappa S., „Performance prediction of component- and pattern-based middleware for distributed systems", 4th on Middleware Doctoral Symposium, Newport Beach, California, 14. számú cikk, 2007. november 26-30.

[23] publikációra hivatkozik:

Woodside M., „The Relationship of Performance Models to Data", Performance Evaluation: Metrics, Models and Benchmarks, Lecture Notes in Computer Science, 5119. szám, 9–28.o., Springer Berlin/Heidelberg, ISBN 978-3-540-69813-5, 2008.

7. Hivatkozott irodalom

[Bernardi 2002] Bernardi S., Donatelli S. és Merseguer J., „From UML Sequence Diagrams and Statecharts to Analysable Petri Net Models", ACM International Workshop Software and Performance, Olaszország, Róma, 35–45.o., 2002.

[Bernardo 1998] Bernardo M. és Gorrieri R., „A Tutorial on EMPA: A Theory of Concurrent Processes with Nondeterminism, Priorities, Probabilities and Time", Theoretical Computer Science, 202. szám, 11–54.o., 1998.

[Bobbio 2004] Bobbio A., Horváth A. és Telek M., „The scale factor: a new degree of freedom in phase-type approximation", Elsevier, Performance Evaluation, 56. szám, 121-144.o., 2004. március.

[Carmona 2002] Carmona D., „Programming the Thread Pool in the .NET Framework", http://msdn.microsoft.com/en-us/library/ms973903.aspx, .NET Development (General) Technical Articles, Microsoft Spanyolország, 2002.

[Ferrari 1972] Ferrari D., „Workload Characterization and Selection in Computer Performance Measurement", Computer, 5(7).szám, 18–24.o., 1972.

[Gilmore 1994] Gilmore S. és Hillston J., „The PEPA Workbench: A Tool to Support a Process Algebra-Based Approach to Performance Modelling", 7th International Conference Modelling Techniques and Tools for Performance Evaluation, 353–368.o., 1994.

[Herzog 2000] Herzog U., Klehmet U., Mertsiotakis V. és Siegle M., „Compositional Performance Modelling with the TIPPtool", Performance Evaluation, 39. szám, 5–35.o., 2000.

[Jain 1991] Jain R., „The Art of Computer Systems Performance Analysis", John Wiley and Sons, 1991.

[Kijima 1997] Kijima M., „Markov Processes for Stochastic Modeling", Chapman & Hall/CRC, 1997.

[Kleinrock 1975] Kleinrock L., „Theory, Volume 1, Queueing Systems", John Wiley and Sons, 1975.

[Marquardt 2003] Marquardt T., „ASP.NET Performance Monitoring, and When to Alert Administrators", http://msdn.microsoft.com/en-us/library/ms972959.aspx, ASP.NET Technical Articles, 2003.

[Meier 2004] Meier J.D., Vasireddy S., Babbar A. és Mackman A., „Improving .NET Application Performance and Scalability (Patters & Practices)", Microsoft Corporation, 2004.

[Menascé 2000] Menascé D.A. és Almeida V.A.F., „Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning", Prentice Hall, 2000.

[Menascé 2001] Menascé D.A. és Almeida V., „Capacity Planning for Web Services: Metrics, Models, and Methods", Prentice Hall PTR, 2001.

[MsPortal] Microsoft Felsőoktatási Portál: http://www.msportal.hu.

[Platt 2003] Platt D.S., „Introducing Microsoft .NET, Third Edition", Microsoft Press, 2003.

[Reiser 1980] Reiser M. és Lavenberg S.S., „Mean-Value Analysis of Closed Multichain Queuing Networks", Association for Computing Machinery, 27. szám, 313–322.o., 1980.

[Sinclair 2005] Sinclair B., „Mean-Value Analysis, Computer Systems Performance Handout, 2005.

[Sopitkamol 2005] Sopitkamol M. és Menascé D.A., „A Method for Evaluating the Impact of Software Configuration Parameters on E-commerce Sites", ACM 5th International Workshop on Software and Performance, Spanyolország, Palma, Illes Balears, 53–64.o., 2005.

[Trivedi 1982] Trivedi K.S., „Probability and Statistics with Reliability, Queueing and Computer Science Applications", Prentice-Hall, 1982.

[Urgaonkar 2005] Urgaonkar B., „Dynamic Resource Management in Internet Hosting Platforms", Ph.D. disszertáció, Massachusetts, 2005.

[Wienholt 2003] Wienholt N., „Maximizing .NET Performance", Apress, 2003.

[Zahorjan 1982] Zahorjan J., Sevcik K.C., Eager D.L. és Galler B., „Balanced Job Bound Analysis of Queueing Networks", Communications of the ACM, 25(2). szám, 134–141.o., 1982.