SZŰK KERESZTMETSZETEK FELTÁRÁSA TÖBBRÉTEGŰ ARCHITEKTÚRÁKBAN

FINDING BOTTLENECKS IN MULTI-LAYERED SYSTEMS

Boros János[1], Horváth Ádám[2] és Jereb László[3]

Összefoglaló: A többrétegű informatikai alkalmazások szűk keresztmetszeteinek feltárása a mai nagy informatikai rendszerek egyik legkritikusabb kérdése. Ennek során tudni szeretnénk, hogy az egyes folyamatok mennyi memóriát és CPU időt használnak, illetve mennyi lemezműveletet igényelnek. Az eszközök terhelési jellemzőiből következtethetünk arra, hogy melyik rétegben és mi okozza a rendszer lassulását, leállását, illetve hogy mik azok a speciális helyzetek, amelyek alkalmazási problémákhoz vezetnek (D. A. Menascé és V. A. F. Almeida 2002).

A rendszerek folyamatainak monitorozására a piacon ingyenes és fizetős eszközök egyaránt elérhetők, tapasztalataink szerint azonban egyik sem kínál olyan megoldást, mellyel a rendszer fizikai eszközeinek valamennyi számunkra fontos paraméterét (CPU, memória és I/O értékek) folyamatokra lebontva határozhatnánk meg. Sok esetben kinyerhetőek a szükséges adatok az operációs rendszerből, de nem folyamatokra lebontva (*iostat*), más esetben folyamat szintű adatok érhetőek el, de nem pontosan azok a mérőszámok, melyekre a vizsgálatunk koncentrál (*top*, *iotop*).

A hiányosságok felszámolására saját eszközt fejlesztettünk, amely a Linux fájlrendszeréből kiolvasott adatokra támaszkodva szolgáltatja a kívánt paramétereket. Vizsgálataink alapján a jövőben azokat a helyzeteket szeretnénk meghatározni, amelyek esetén a rendszer túlterhelése, esetleg összeomlása várható. E helyzetek azonosítása lehetővé tenné, hogy a rendszer előre tudjon reagálni a kritikus időszakokra.

Kulcsszavak: többrétegű architektúra, szűk keresztmetszet, rendszer monitorozás.

Abstract: Finding the bottlenecks of a multi-layered applications system is one of the most critical issues of large information systems. In these cases we want to know how much memory, CPU slots and disk operation are used by the processes. From the load parameters the reasons of the slowdown or system crash in any system layer can be determined, and furthermore, the special situations resulting in application problems can be identified.

Several free and commercial tools are available for the monitoring of system processes; however, our experience shows that none of them can provide all process parameters that are important for us (CPU, memory and I/O), or in some cases, the aggregated data can be obtained from the operating system and not the per process values (*iostat*), or conversely, per process data are available, and not the necessary system parameters (*top, iotop*).

To handle these problems, we developed a software that can monitor the above mentioned parameters per process, based on the file system of the Linux operating system. Using this software our objective is to build up a proactive system that is able to identify the situations leading to overload or system crash, and therefore, that makes it possible to avoid them.

Keywords: multi-layered system, bottlenecks, system monitoring.

1. Bevezetés

Sok rendszer esetén érzékelhető, hogy hosszabb-rövidebb ideig nem tud a kívánt szinten szolgáltatni. A probléma gyakran a túlterhelésre vezethető vissza, mivel a rendszer felkészült az

[1] Nyugat-magyarországi Egyetem, Informatikai és Gazdasági Intézet, borosj@inf.nyme.hu
[2] Nyugat-magyarországi Egyetem, Informatikai és Gazdasági Intézet, horvath@inf.nyme.hu
[3] Nyugat-magyarországi Egyetem, Informatikai és Gazdasági Intézet, jereb@inf.nyme.hu

átlagos terhelésre, azonban már a rövidebb ideig tartó, megnövekedett terhelést sem tudja kezelni, ezért összeomlik. Kisméretű rendszerek, szoftverek vagy könnyen átlátható normál asztali PC esetén az ok viszonylag egyszerűen kideríthető. Nagyméretű, bonyolult felépítésű szerverek vagy sok esetben szerverfarmok esetén azonban már nem triviális annak meghatározása, hogy mi és miért történt. Egyszerű példaként a Felvi felvételi rendszere, amely egész évben megfelelően működik, a jelentkezési határidők környékén azonban többször volt tapasztalható olyan anomália, amikor a rendszer elérhetetlenné vált. Célunk ezért, hogy az ilyen rendszerek előzetes elemzése alapján meg tudjuk mondani, hogy milyen módon lehet minimális anyagi ráfordítás mellett, hirtelen megugró terhelés esetén is teljesíteni a követelményeket.

Az ismertetett eredmények egy innovációs ipari projekt keretében születtek. Megbízónk a budapesti székhelyű Netvisor Zrt. A többéves projekt keretében a vizsgálatok és fejlesztések irányát a megbízó piaci igényei és szakmai tapasztalatai határozták meg, egyaránt kihatva a vizsgált rendszerek működési környezetére (PVSR), a vizsgált adatbázis-kezelő rendszerekre, valamint az elvárt vizsgálati eredményekre. A követelmények közül kiemelendő, hogy az eszközökre és folyamatokra vonatkozó jellemzőkre egyaránt szükség van, miközben a hatékony alkalmazhatóság érdekében célszerű elkerülni az operációs rendszerek alapszolgáltatásain túli szoftverek telepítését.

2. Módszerek és eszközök

A 2010 tavaszától, 2011 végéig terjedő K+F folyamatot négy nagyobb, jól elkülöníthető részre bontottuk. Első körben feltérképeztük a rendelkezésre álló eszközöket, majd létrehoztunk egy mérőszervert adott rendszer alá (PVSR). Ezt követi az adatréteg monitorozása: különböző adatbázis-kezelők viselkedését térképezzük fel, majd a projekt zárásaként egy modult fejlesztünk korábbi tapasztalatainkra építve a PVSR alá, melynek feladata, hogy IBM Informix, PostgreSQL illetve Sybase adatbázis-kezelőkről szolgáltasson információkat. Az első és második szakasz már lezárult, pillanatnyilag a harmadik ponttal foglalkozunk.

2.1. Linux eszközök értékelése

A Linux operációs rendszer teljesítmény elemzéséhez olyan standard és speciális (egyedi, kísérleti) eszközöket keresünk, amelyek pontos és hasznos információkat szolgáltatnak a Linux kernel felett futó folyamatok memória, CPU, diszk terhelési adatairól, és amelyek segítségével összefoglaló képet lehet kapni a rendszer állapotáról. Esetünkben fontos szempont, hogy a későbbi elemzés céljából egy adott eszköz képes-e folyamatos megfigyelésre és az idősor adatok lemezre mentésére. Ennek figyelembevételével módszert dolgoztunk ki a különböző mérőeszközök összehasonlítására. Minden eszközt futás közben vizsgáltunk és megnéztük, hogy melyikkel milyen paraméterek nyerhetők ki. A fő szempontok: vizsgált paraméterek, folyamatszintű monitorozás, időbélyeg megléte és a további felhasználhatóság miatt a licensz kérdése voltak (Boros et al 2010).

Az Unix rendszer a működési információkat a /proc fájlrendszer alá helyezi. Ez egy speciális fájlrendszer, amely a diszken nem jelenik meg, csak a memóriában található. A /proc alatt mind a rendszer hardveres, mind pedig a szoftveres működési paraméterei naplózásra különböző fájlokban kerülnek, amely paraméterek később kiolvashatók és feldolgozhatók.

Első lépésben a standardnak számító vmstat, iostat, top, ps, stb. monitorozó programok képességeit és a rendszeranalízis szempontjából hasznos szolgáltatásait tekintettük át, ezen kívül olyan ingyenesen felhasználható eszközöket kerestünk, amelyek a fentieken túl további hasznos információkat tudnak nyújtani. Mivel mindegyik mérési program a /proc fájlrendszer alapján dolgozik, a vizsgált programok csak abban különböznek, hogy milyen paramétereket tartanak fontosnak a programozók és milyen ügyesen valósították meg a programjukat.

A mérések és tapasztalatok alapján a dstat széles körben és könnyen használható, mivel segítségével könnyen lehet mérni a rendszer CPU, memória, I/O és egyéb paramétereinek változásait. Ugyanakkor alkalmazását nagy mértékben korlátozza, hogy használatához különböző segédprogramok és egyéb mérést segítő eszközök telepítésére van szükség. Ez sok esetben a valós használat során nem tehető meg. Mivel mi egy kompakt, könnyen kezelhető és tisztán az operációs rendszer szolgáltatásaira támaszkodó eszközt kerestünk, a dstat nem jó választás.

A második vizsgált eszköz az iotop volt. Az iotop alapesetben a rendszer egészét, tehát az összes lemezt illetve folyamatot tekintve az összes olvasott illetve írt adat mennyiségét szolgáltatja. Ennek során megadja a folyamat azonosítóját (PID), a folyamatot futtató felhasználót (USER), folyamatonkénti bontásban az írt és olvasott adatmennyiséget byte/s-ban, továbbá megtudjuk, milyen arányban foglalkozott a folyamat (szál) swap műveletekkel illetve milyen arányban várt a folyamat (szál) I/O műveletekre. A parancs folyamatosan fut, mód van futási időben változtatni néhány jellemzőt. Adott időközönként frissíti a listát a folyamatokról, amivel értesülünk a rendszer változásairól. A fenti előnyök mellett az iotop fő hátránya, hogy alap Linux disztribúciókon nem lehet használni az alkalmazást, mivel a megfelelő működéshez további csomagok telepítése szükséges, ezért választása ugyancsak kizárható.

A harmadik tételesen megvizsgált eszköz az iostat volt, amelynek segítségével CPU és diszk műveleteket is monitorozhatók. A diszkre koncentrálva megállapítható, hogy könnyen paraméterezhető, így egyszerűen lehet a riportot szűkíteni a számunkra fontos eszközökre. Ez a szoftver tehát nem a folyamatokkal, hanem az eszközökkel foglalkozik, így azonosítja a gépet, az operációs rendszer verzióját, a processzorok számát, típusát. Egy eszközhöz megadja a másodpercenkénti transzferek számát, az olvasott és írt adatok mennyiségét (paramétertől függően blokkban, kilobyte-ban vagy megabyte-ban). Alapértelmezésben egyszeri mintavételezésről van szó. Ha azt szeretnénk, hogy több mérés történjen, akkor ez további paraméterezéssel biztosítható.

Az utóbbi két alkalmazást összehasonlítva elmondható, hogy az iotop inkább a processzenkénti bontásra koncentrál, és nehezebb összegző adatokat kinyerni segítségével a rendszer egészéről. Az iostat ezzel szemben az eszközönkénti terhelést tükrözi, tükrözi, de a folyamatokról és a gyakran fontos SWAP paraméterekről csak korlátozottan képes információt szolgáltatni.

Az ugyancsak vizsgált ingyenes collectl parancs kétféle módban gyűjthet adatokat a rendszer kihasználtságáról. Record módban a /proc könyvtárból dolgozik, kimenete a képernyő vagy egy fájl lehet, míg playback módban az általunk generált adatfájlokból dolgozik. Az egyik legösszetettebb szoftver, amit Linux környezetben, monitorozásra lehet használni, de csakúgy, mint az iotop esetén, ez sem használható kiegészítő csomagok nélkül, így alkalmazása számunkra kizárható.

Végrehajtottunk egy olyan méréssorozatot is, amelyben mindhárom mérőszkripttel, ugyanazon terhelés mellett, azonos környezetben mértünk. A mérés során fájlok létrehozásával és másolásával terheltük a rendszert. Egyrészt kisméretű, másrészt nagyméretű fájlokkal dolgoztunk, és figyeltük, hogy melyik esetben mennyire terhelődik a környezet, amivel egy további szemszögből is vizsgálni tudtuk az eszközöket.

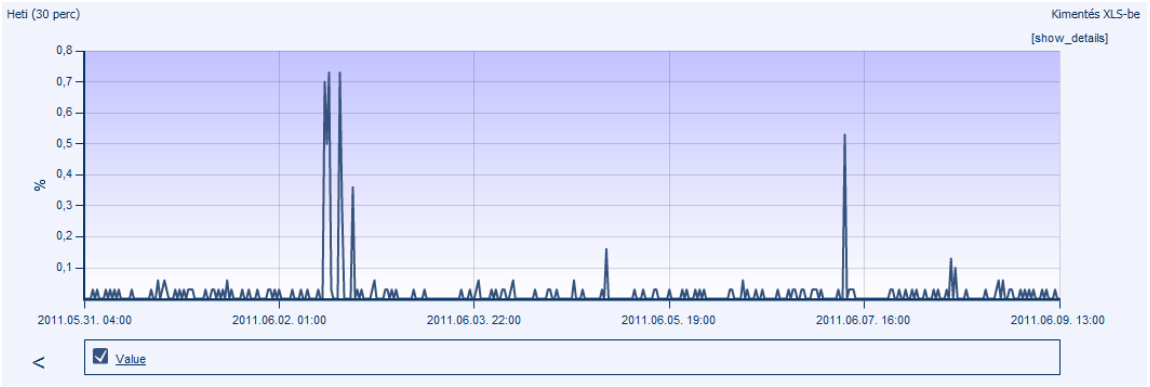
A rendelkezésre álló alkalmazások széleskörű vizsgálatával nagyszámú információt gyűjtöttünk össze a rendelkezésre álló és használható szoftverekről. Számos nagyon hasznos és jól működő szoftver található, de minden esetben adódtak problémák is: vagy nem megfelelő paramétert monitoroztak vagy nem a megfelelő mélységben, ezért úgy ítéltük meg, hogy speciális követelményeink kielégítésére saját fejlesztésű környezetet célszerű létrehozni.

2.2. Saját fejlesztés

A fenti mérőalkalmazások tapasztalatainak felhasználásával olyan saját alkalmazást hoztunk létre, amely már megfelel az elvárásainknak. A fejlesztés során figyelembe kellett vennünk, hogy az alkalmazás egy meglévő rendszer (PVSR) alatt kell, hogy működjön (Jereb et al 2011).

Az alkalmazás egy PERL nyelven megírt, Linux alatt futtatható program. Bemeneti paraméterként vár egy karaktersorozatot, ezt illeszti a mérendő rendszer futó folyamatainak nevére és így szűkíti a monitorozni kívánt szálak számát illetve körét. A mérés gyakoriságát a futtató környezet, azaz a PVSR vezérli. Alapvetően 7 különböző rendszerparamétert vizsgálhatunk a segítségével: a karaktersorozatra illeszkedő folyamatok számát, a CPU usr értékét, a CPU sys értékét, a használt fizikai memória mennyiségét, a használt swap memória mennyiségét, továbbá a folyamatok által olvasott illetve írt adatmennyiséget.

Miután megtörtént a mintavétel, az adatokat Oracle adatbázisban tároljuk és az előző mért értékekhez viszonyítva számolunk különbségeket. Ezeket az értékeket grafikonon ábrázoljuk, webes felületen (1. ábra).



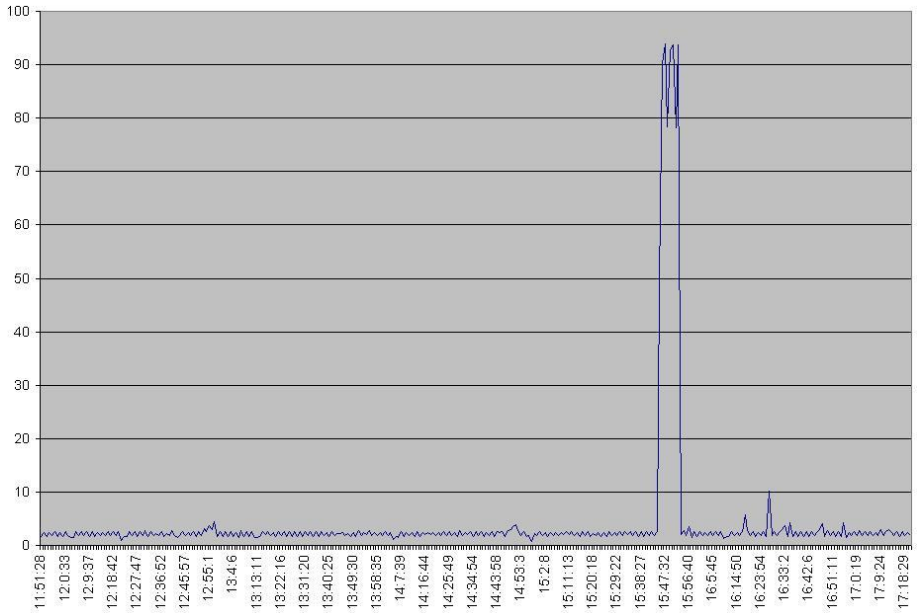
1. ábra Átlagos CPU terhelés 30 perces bontásban

Az alkalmazás hitelességének igazolására a mért eredményeket a korábban tárgyalt collectl és top eszközök által rendszerből kiolvasott adatokkal hasonlítottuk össze. Az összehasonlítás során egyazon rendszert teszteltük és terheltük, figyelve, hogy ugyanazon időpillanatban milyen értékeket szolgáltat egyrészt az általunk írt alkalmazás, másrészt a külső szoftver.

Az alábbi ábrákon (2. és 3. ábra) látható a két szkript által szolgáltatott mérési eredmények összehasonlítása. Látható, hogy a rendszer CPU terhelése alapvetően néhány százalékot tesz ki (2-3%), míg rövid ideig terheltük a rendszert a speciálisan erre a célra fejlesztett szkripttel, amikor 90% körüli eredményeket tapasztaltunk.

Ebben az esetben a választott hitelesítő eszköz a top volt, a két grafikon időskáláján látható különbség abból fakad, hogy míg a szkriptünk által mért adatokból a PVSR generál böngészőben megjelenített grafikonokat, addig a top által mért adatokból nekünk kellett manuálisan grafikont létrehozni.

2. ábra CPU terhelés az általunk fejlesztett szkript alapján



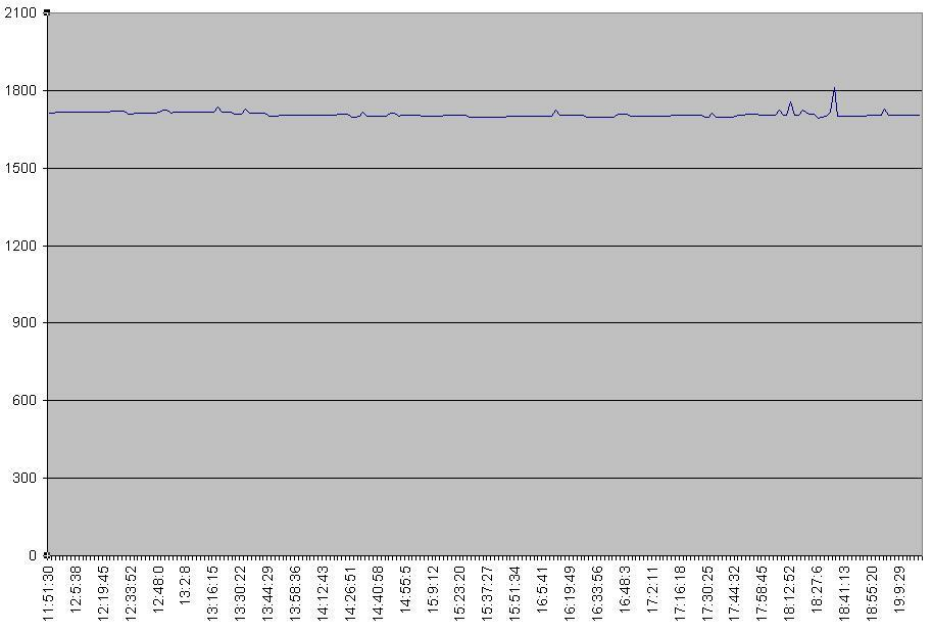
3. ábra CPU terhelés a top alapján

Mivel több paramétert vizsgálunk egy adott mérés során, nem elegendő a CPU értékek összehasonlítása. A következő négy ábrán (4., 5., 6. és 7.) a memória használatának mért értékei láthatóak, elsőként a saját alkalmazás majd a külső program által mért adatokkal és grafikonnal. Ebben az esetben nem a top-ot használtuk hitelesítésre, hanem a collectl nevű alkalmazást.

Memória esetén a hitelesítés során külön monitoroztuk a folyamatok által használt fizikai memória méretét illetve a swap, azaz az átlapolt memória használatot. Ez az a memóriaterület, melyet a folyamatok közösen, megosztva használnak.

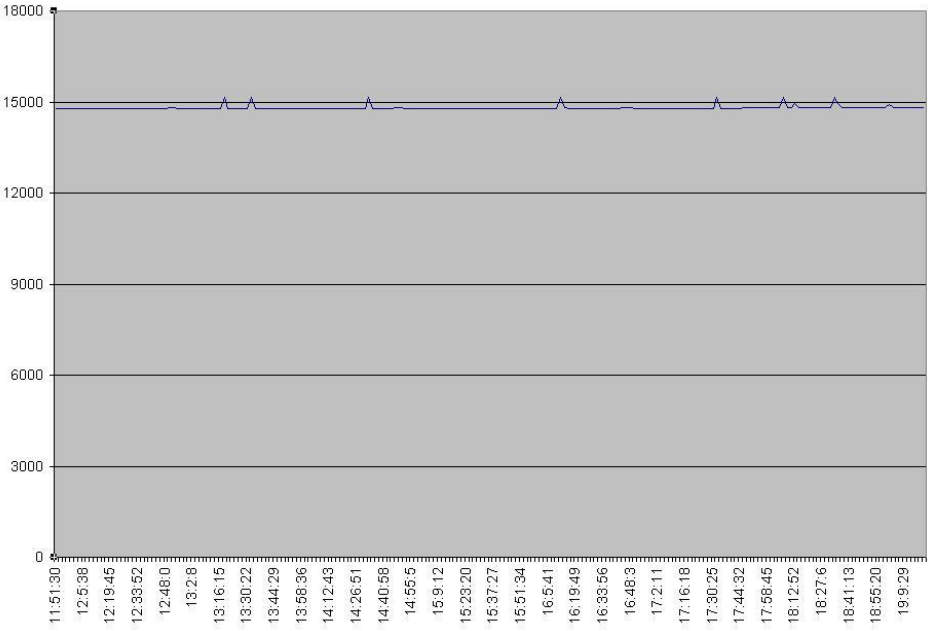
Összefoglalva megállapítható, hogy a saját a létrehozott saját alkalmazás minden mérési paraméter esetén összhangban van az elérhető és vizsgált, piacon elérhető más eszközökkel.

4. ábra Memória (rss, swap) terhelés az általunk fejlesztett szkript alapján



5. ábra Memória (rss, swap) terhelés collectl alapján

6. ábra Memória (vsz, fizikai) terhelés az általunk fejlesztett szkript alapján



7. ábra Memória (vsz, fizikai) terhelés collectl alapján

3. Elért eredmények

A munkánk és kutatásunk eredményeként egy megbízhatóan működő mérőszervert alakítottunk ki egy meglévő, PVSR rendszer alá. Az általunk fejlesztett alkalmazás megbízhatóan szolgáltat adatokat tetszőleges Linux operációs rendszert futtató környezetből. A szkript segítségével tetszőleges folyamat mérőszámai kinyerhetőek a rendszerből, segítségével képet kaphatunk a CPU, a memória és az I/O jellemzők értékéről. Egy időpontban több eszköz mérése is futhat párhuzamosan, illetve egy eszközön több mérési indexszel rendelkező mérési folyamatot is tudunk egy időben futtatni. Így egy szervert tekintve, egyszerre tudjuk monitorozni az összes diszk I/O mutatóit, és bármely diszken vizsgálni tudjuk azt is, hogy mennyire terhelik a különböző folyamatok (pl. Oracle folyamatok, MySQL folyamatok).

A modul kialakítását tekintve teljes mértékben moduláris, ami lehetővé teszi, hogy a későbbiekben könnyen lehessen bővíteni tetszőleges paraméter monitorozási lehetőséggel.

Az elkészült modul a megfelelő tesztelési és ellenőrzési folyamatok után piaci környezetben hasznosításra kerül, és a PVSR mérőeszköz következő verziójában elérhető szolgáltatásként jelenik meg.

4. Összefoglalás

Az elvégzett munka során Linux környezetben nagyon sok ingyenes és fizetős eszközt vizsgáltunk részletesen, amely eszközök segítségével különböző információkat nyerhetünk ki adott rendszerekből. Mivel viszonylag speciális igényeink voltak a fejlesztendő alkalmazást tekintve, egyértelművé vált, hogy azok kielégítésére saját alkalmazás létrehozására van szükség.

A következő lépésben elkészült ez az alkalmazás, amelyről a tesztelések során bebizonyítottuk, hogy PVSR környezetben egyértelműen alkalmas a kívánt funkciók megvalósítására.

A projekt következő szakaszában a súlypont az adatbázis-kezelő rendszerek belső felépítésének és viselkedésének részletes elemzése következik, majd a mérőmodult fejlesztjük tovább, ugyancsak a PVSR alá, annak piaci funkcióinak bővítésére.

5. Köszönetnyilvánítás

A szerzők ez úton is köszönetet mondanak Máthé Jánosnak, Schnell Györgynek, Szabó Balázsnak és Szemethy Tivadarnak, a Netvisor vezetőinek, illetve munkatársainak a feladat megfogalmazásáért és a megoldás során adott értékes tanácsaikért.

Irodalomjegyzék

D. A. Menascé, V. A. F. Almeida (2002) Capacity planning for Web services. Prentice Hall

Boros J., T. V. Do, Horváth Á., Szalai L. (2010) Linux rendszerek teljesítményanalízise. RET kutatási jelentés, NyME FMK INGA

Jereb L., T. V. Do, Boros J., Horváth Á. (2011) Mérőszerver fejlesztése PVSR alá. RET kutatási jelentés, NyME FMK INGA