Nyílt forrású webkiszolgálók

Ibrahim három nyílt forrású webkiszolgáló teljesítményét próbálja ki az Ericsson Research telephelyén használt Linux-fürtben.

A kanadai Ericsson Researchnél 2000. januárjában indult az ARIES (Advanced Research on Internet E-Servers) projekt. Célja az volt, hogy egy a Linuxra és nyílt forrású programokra épülő, összetett telekommunikációs feladatok ellátására képes kiszolgálófürt képességeinek bizonyításához szükséges módszereket fejlesszünk ki.
A linuxos fürtök iránt támasztott telekommunikációs követelmények igen szigorúak és a szakágban mindenki jól ismeri az ehhez kapcsolódó nehézségeket. Itt ugyanis biztos elérhetőségre (a rendszernek a hét minden napján, a nap minden órájában elérhetőnek kell lennie), garantált válaszidőre (statisztikailag szavatolt legnagyobb késedelemre), méretezhetőségre és garantált teljesítményre van szükség (a rendszernek képesnek kell lennie arra, hogy időegység alatt legalább egy meghatározott mennyiségű kérelmet feldolgozzon).
Mindemellett a telekommunikációs internetes kiszolgálóknak további fontos feltételeknek is meg kell felelniük, például az internetes forgalom ugrásszerű (félévente százszázalékos) növekedésének, azonban említhetnénk a felhasználók minőségbeli elvárásainak növekedését vagy az igen magas biztonsági követelményeket is.
Ezek az internetes kiszolgálók nagyteljesítményű és nagymértékben méretezhető webkiszolgálókat igényelnek. Mivel az ARIES-nél végzett munka a nyílt forrású programokra épül, már csak egy olyan nyílt forrású webkiszolgáló hiányzott, amely lehetővé tette volna rendszerünk kiépítését.
Az ARIES-nél az egyik célunk az volt, hogy olyan internetes kiszolgálót készítsünk, amely a letöltési sebességek észrevehető csökkenése nélkül képes akár több ezer felhasználót is egyidejűleg kiszolgálni. Ezt a fajta méretezhetőséget legkönnyebben kiszolgálófürtök alkalmazásával érhetjük el. Amikor egy adott honlap egy bizonyos oldalára irányuló kérelem érkezik, azt a legkevésbé terhelt kiszolgálóhoz irányítjuk (egy okos forgalomelosztó megoldással, mely lehet gépi vagy programból megvalósított is).
Három kiszolgáló: az Apache, a Jigsaw és a Tomcat kipróbálása mellett döntöttünk. Az Apache a világ legnépszerűbb webkiszolgálója, az ARIES 2000. évi indulása óta kísérletezünk vele. A Java-alapú Jigsaw jelenleg a kísérleti Linux-fürtön használatos. A Tomcat is Java-alapú webkiszolgáló, a jövőben talán a Jigsaw helyére léphetne, amennyiben nagyobb teljesítményűnek bizonyul.
Az Apache webkiszolgáló hatékony, rugalmas, az előírásainak megfelelő HTTP 1.1 webkiszolgáló. A Netcraft Web Servers felmérése alapján az Apache 1996 áprilisa óta az Internet legnépszerűbb webkiszolgálója. Ez nem meglepő, ha számos előnyös tulajdonságára gondolunk: több felületen használható, megbízható, jól beállítható és forráskódja bárki számára hozzáférhető. Próbáink során az Apache 1.3.14-gyel (ez akkoriban a legfrissebb megbízható változat volt) és az Apache 2.08 alpha változattal kísérleteztünk.
A Jigsaw a W3C nyílt forrású projektje, mely 1996 májusában indult. Ez egy webkiszolgálófelület, mely a HTTP 1.1 bemutató jellegű támogatását és számos további lehetőséget tartalmaz, és egy Javában készített fejlett rendszerre épül.
A Jigsaw-t nem nyilvános megjelenésre, sokkal inkább a különféle módszerekkel való kísérletezés céljából készítették. Próbáinkban a Jigsaw 2.0.1-et és a Java 2 SDK-t használtuk.
A Jigsaw a 8001-es kapunk fogadta a http-kérelmeket.
A Tomcat a Java Servlet 2.2 és a JavaServer Pages 1.1 referenciamegvalósítása. Az Apache felhasználási szerződése alatt fejlesztett Tomcat egy JSP-környezettel rendelkező servletburok, azaz egy futásidejű héj, mely a servleteket a felhasználók nevében kezeli és hívja meg. A Tomcatet önálló kiszolgálóként vagy egy másik webkiszolgáló, például az Apache bővítményeként használhatjuk. Próbáinkban a Tomcat 3.1-es változatát telepítettük önálló kiszolgálóként, mely a kérelmeket a 8080-as kapun fogadta.

1. kép Egy jellegzetes Linux-fürt az Ericsson Research laborjában

A Linux-fürt beállítása

A fentebb említett webkiszolgálók kipróbálása és összehasonlítása érdekében felállítottunk egy az Ericsson Researchnél általánosan használt linuxos fürtöt (*1. kép*).
Ezt a felületet csúcsteljesítményű kiszolgálóalkalmazások számára terveztük. A próbakörnyezet az alábbi elemekből épült fel:

- Nyolc merevlemez nélküli, 500 MHz-en futó és 512 MB memóriával felszerelt Pentium III CompactPCI processzorkártya. A processzorok két felületszerelt hálózati csatlakozóval rendelkeznek, s ehhez jön még egy négycsatlakozós ZNYX ethernetkártya, melyek együttesen több hálózati kapcsolatot tesznek lehetővé.
- Nyolc számítógép ugyanezzel a felépítéssel. Az egyetlen különbség, hogy ezek a gépek egy közös merevlemezblokkal is rendelkeznek, mely három, a legjobb elérhetőség érdekében RAID 1 és RAID 5 rendszerbe kötött 18 GB-os SCSI-merevlemezből állt.
- Fő elemek: a lemezes gépek közül kettő egymás felváltására képes NFS-, NTP-, DHCP- és TFTP-kiszolgálóként működik a többi gép számára. Az egymás kiváltását lehetővé tevő kódot saját magunk fejlesztettük ki, csakúgy, mint a két NFS-kiszolgáló ugyanazon csatolási pontra történő befűzését lehetővé tevő különleges `mount` programot.

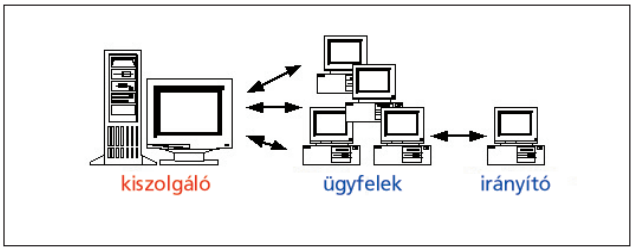
2. kép A próbaegységek

A gépek a helyi hálózatról indulnak el (vagy az 1-es, vagy pedig a 2-es LAN-ról, attól függően, hogy épp melyik működik), majd DHCP-kérelmet intéznek a hálózat minden IP-címe felé. A fő elemek DHCP-választ küldenek és egyúttal továbbítják a gépeknek a hálózati adataikat, így az IP-címek (minden csatoló, az eth0, az eth1, a znb1 és a znb1 számára egy-egy cím), az átjáró, a hálózati maszk, a tartománynév, az indítókiszolgálók IP-címei és az indítófájl nevének beállításához szükséges adatokat. A lemez nélküli gépek ezután letöltik és elindítják a DHCP-beállításfájlban meghatározott indítófájlt, mely a DHCP-kiszolgáló */**tftpboot*** könyvtárában található rendszermagfájl. A következő lépésben a gépek betöltenek egy ramlemezt és indítják az alkalmazáskiszolgálókat, azaz az Apache-t, a Jigsaw-t és a Tomcatet. Egy lemez nélküli kiszolgáló elindítása (az indítás másodpercétől a készenléti jel megjelenéséig) kevesebb mint egy perc alatt lezajlik.
A lemezes gépek szintén a DHCP-beállításfájlban meghatározott indítófájlt töltik le és indítják el (ez a rendszermagfájl is a DHCP-kiszolgáló */**tftpboot*** könyvtárában található). Ezt követően megtörténik a RAID önműködő beállítása és a Red Hat 6.2 testreszabott telepítése. Amikor a gépek működésre készen állnak, elindítják az Apache-, Jigsaw- és Tomcat-kiszolgálókat, mindegyiket külön kapun. A lemezes gépek üzemkész állapotba hozása nagyjából öt perc alatt történik meg (ebbe a RAID1 és a RAID önműködő beállítását, illetve a RedHat 6.2 telepítését is beleszámoltuk).
Próbáink számára a lemezes gépeket (pontosabban közülük hatot, tehát a fő elemeket nem) szintén lemez nélküli gépként indítottuk el az egységes próbakörnyezet kialakítása érdekében.

A próbakörnyezet

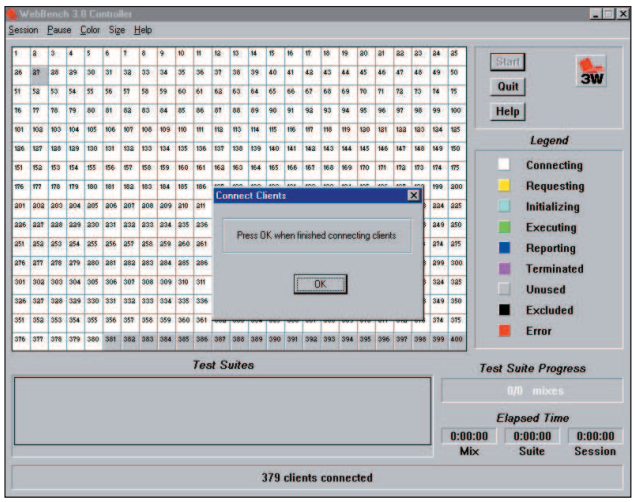
A webkiszolgálók és a kiszolgálóalapú rendszerek teljesítményét számos tényező befolyásolja, például az ügyfél által használt felület és alkalmazás, a kiszolgálón használt felület és alkalmazások, valamint a hálózat és az azon használt protokollok. A Világhálót elemző teljesítménytesztek általában két dologra összpontosulnak: az általános hálózati teljesítményre és a webkiszolgáló alkalmazás- és felület-teljesítményére.
Próbáinkban egy webkérelmeket létrehozó, szabványos mérések végzésére alkalmas módszert használtunk. A forgalmat a WebBench nevű ingyenes programmal (letölthető a ➲ http://www.zdnet.com címről) 16 darab 500 MHz-es, 1 keretheletyet elfoglaló (1U méretű) Intel Celeron gép állította elő (*2. kép*).

Az alap-próbakörnyezetben ügyfélprogramok egy csoportja kérelemfolyamot továbbított a kiszolgálók felé és mérte a rendszer válaszát. A kérelemfolyamot terhelésnek nevezzük. A WebBench a webkiszolgálók teljesítményének mérésére alkalmas módszert biztosít, egy vezérlőből és számos ügyfélprogramból áll (*3. kép*). A vezérlő a WebBench-tesztek beállítását, indítását, leállítását és figyelését végzi, valamint az ügyfélprogramoktól érkező adatok összegyűjtéséért és elemzéséért is felelős.



3. kép A WebBench felépítése

A másik oldalon az ügyfelek elindítják a WebBench-teszteket és kérelmeket küldenek a kiszolgálóhoz. A WebBench az ügyfélgépek használatával a böngészőprogramokat utánozza.
A valódi böngészőkkel ellentétben a kiszolgálóról válaszként kapott fájlokat nem jelenítik meg. Ehelyett amikor egy ügyfélprogram választ kap a kiszolgálótól, akkor a válasszal kapcsolatos minden adatot rögzít, majd azonnal újabb kérelemmel bombázza a kiszolgálót.
A webkiszolgálók teljesítményét sokféleképpen mérhetjük. Próbánkban a kiszolgáló által egy másodperc alatt teljesített kérelmek számát, illetve az egy másodperc alatt küldött bájtok számát rögzítjük (*4. ábra*).



4. kép A WebBench vezérlőablaka

A WebBench a kiszolgáló teljesítményét szabványos próbacsomaggal méri. A próbacsomag tömörített formában szerezhető be, melyet saját magunknak kell kibontanunk a webkiszolgáló sajátkönyvtárában (a webkiszolgáló itt keres minden HTML- és egyéb fájlt). Így létrejön a WBTREE nevű könyvtár, mely 61 MB webdokumentumot tartalmaz, melyeket a WebBench-ügyfelek kérni fognak. Mivel néhány gép nem tartalmaz merevlemezt, a terhelési csomagot az NFS-kiszolgálóra telepítettük, s a webkiszolgálót úgy állítottuk be,

hogy a webdokumentumok alapértelmezett keresési útvonala az NFS-kiszolgáló megfelelő könyvtára legyen.
A WebBench beállításának részeként azt is beállítottuk, hogy a próbagépek által gerjesztett forgalom a kiszolgálókon egyenlő mértékben osztódjon szét. Az *5. kép* azt mutatja be, hogy miként állítottuk be az egyes kiszolgálóelemeket és az általuk fogadott forgalmi százalékokat.

CPU	%
CPU 1	25%
CPU 2	25%
CPU 3	25%
CPU 4	25%

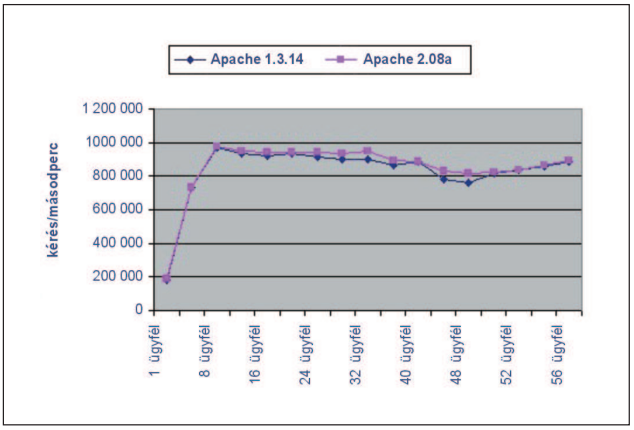
5. kép Egy példa a WebBench beállítására

Ügyfelek száma	Apache 1.3.14	Apache 2.08a
1_ügyfél	182,517	187,333
4_ügyfél	731,067	735,783
8_ügyfél	971,900	978,800
12_ügyfél	931,500	946,067
16_ügyfél	922,767	939,133
20_ügyfél	933,217	943,333
24_ügyfél	913,383	939,050
28_ügyfél	903,333	935,717
32_ügyfél	899,467	945,817
36_ügyfél	867,700	894,633
40_ügyfél	885,933	887,933
44_ügyfél	782,283	828,650
48_ügyfél	759,583	816,017
52_ügyfél	817,150	821,700
56_ügyfél	840,217	838,950
60_ügyfél	856,833	864,133
64_ügyfél	883,067	890,366

6. kép Az Apache 1.3.14/2.08a egygépes próbaadatai

A próbák

A Linux-fürt és a próbakörnyezet felállítása után készen álltunk arra, hogy meghatározzuk a kipróbálásra kerülő rendszereket. Mindhárom webkiszolgálót (Apache, 1.3.14 és 2.08a, Tomcat 3.1, Jigsaw 2.0.1) kipróbáltuk 1, 2, 4, 6, 8, 10 és 12 processzoron. Minden próbánál a betöltendő ramlemezen határoztuk meg, hogy a ramlemez betöltése után mely webkiszolgáló legyen elindítva. A fentiek eredményeképpen négyféle próbát hajtottunk végre, mindegyiket különböző kiszolgálóval és több tesztgéppel.
Cikkemben csak három összehasonlító próbaeredményt mutatunk be: az Apache 1.3.14 és a 2.08a összehasonlítását egy gépen, az Apache 1.3.14 és az Apache 2.08a összehasonlítását nyolc gépen, illetve a Jigsaw 2.0.1 és a Tomcat 3.1 összehasonlítását egy gépen.
Az első teszt során az összes webkiszolgálót egyetlen gépen próbáltuk ki. A WebBench beállításában azt határoztuk meg, hogy az összes ügyfél forgalma egyetlen gépre kerüljön.
A *6. képen* látható mérés 64 ügyfélprogram egyidejű működése mellett készült. Az Apache 1.3.14. átlagosan 828, míg az Apache 2.08a 846 kérelmet volt képes feldolgozni egy másodperc alatt. Utóbbi tehát 2,1 százalékos fölénnyel büszkélkedhet. A *7. kép* az Apache 1.3.14 és 2.08a eredményeit mutatja be. Amint látható, a két kiszolgáló csaknem egyforma teljesítményű.



7. kép Az Apache 1.3.14/2.08a egygépes próbaeredményei

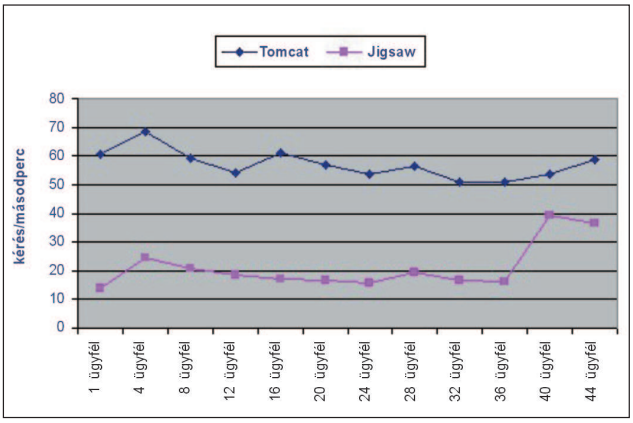
Ügyfelek száma	Tomcat	Jigsaw
1_ügyfél	60,45	14,067
4_ügyfél	68,283	24,296
8_ügyfél	59,283	20,604
12_ügyfél	54,267	18,304
16_ügyfél	60,917	16,908
20_ügyfél	57,067	16,721
24_ügyfél	53,633	15,696
28_ügyfél	56,2	19,533
32_ügyfél	50,866	16,634
36_ügyfél	50,784	16,183
40_ügyfél	53,767	39,358
44_ügyfél	58,816	36,583

8. kép A Tomcat/Jigsaw egygépes próbaadatai

A Java-alapú kiszolgálók (Tomcat és Jigsaw) adatait a *8.* és a *9. képen* láthatjuk. A Jigsaw másodpercenként 39, míg a Tomcat 60 kérelmet volt képes kiszolgálni. A Jigsaw silány teljesítményén nem kell csodálkoznunk, hiszen csupán „kísérleti nyúlnak" tervezték, nem pedig széles körben használt alkalmazásnak. Amikor a próbát nyolc egyidejűleg működő gépre terjesztettük ki, az Apache 2.08a magabiztosabbá vált és az egyidejűleg futó ügyfelek számának növelésével egyre több kérelmet volt képes kiszolgálni, ami a feldolgozott kérelmek számában sem okozott kilengéseket (*10. kép*).
A *11. képen* világosan látszik, hogy ebben a helyzetben az Apache 2.08a mennyivel jobb az 1.3.14-es változatnál. Nyolc gép esetén az Apache 2.08a képes volt megtartani a másodpercenkénti 4434 kérelem–kiszolgálást, míg az Apache 1.3.14 csupán 4152-t teljesített.

A méretezhetőségi próbák eredményei

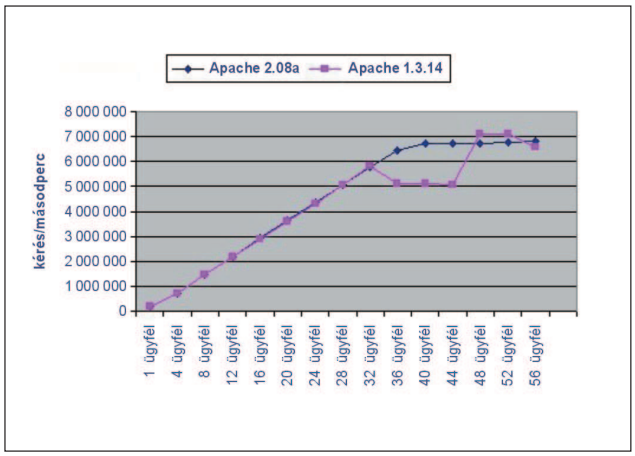
Az 1, 2, 4, 6, 8 és 10 linuxos gépből álló rendszerek grafikonjait gyűjtöttük össze. Minden egyes grafikonnál rögzítettük az adott rendszer által egy másodperc alatt kiszolgált kérelmek legnagyobb számát. Ha ezt elosztjuk a gépek számával, meg-



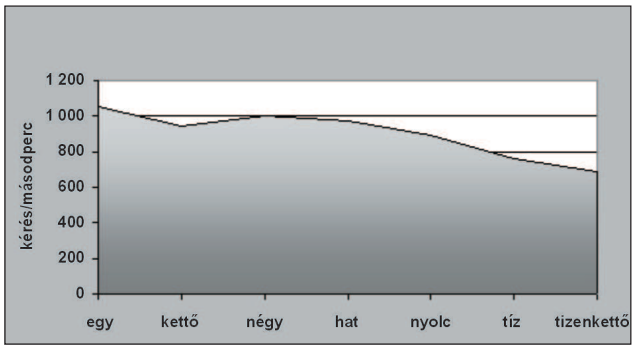
9. kép A Tomcat/Jigsaw egygépes próbaeredményei

Ügyfelek száma	Tomcat	Jigsaw
1_ügyfél	176,417	187,017
4_ügyfél	730,700	728,117
8_ügyfél	1460,483	1446,617
12_ügyfél	2197,383	2162,467
16_ügyfél	2921,216	2869,700
20_ügyfél	3642,583	3598,400
24_ügyfél	4363,667	4300,117
28_ügyfél	5071,800	5088,100
32_ügyfél	5767,833	5821,950
36_ügyfél	6442,083	5124,150
40_ügyfél	6701,150	5119,900
44_ügyfél	6721,317	5085,717
48_ügyfél	6743,100	7092,250
52_ügyfél	6775,300	7092,833
56_ügyfél	6808,617	6570,800

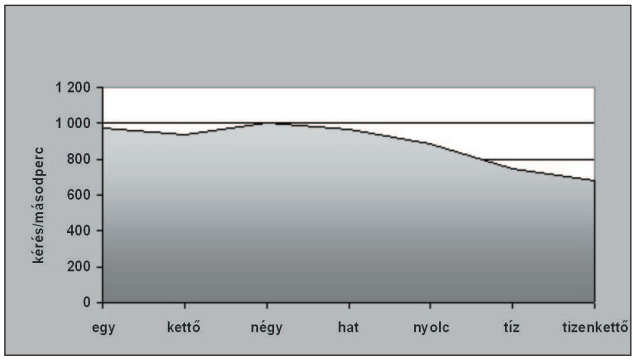
10. kép Az Apache 1.3.14/2.08a nyolcgépes próbaadatai



11. kép Az Apache 1.3.14/2.08a nyolcgépes próbaeredményei



12. kép Az Apache 2.08a méretezhetőségi táblázata



13. kép Az Apache 1.3.14 méretezhetőségi táblázata

kapjuk, hogy az egyes összeállításokban egy gép mekkora teljesítményre képes.

A *12.* és *13. képek* a két Apache-változat gépenkénti átviteli teljesítményét mutatják be, szembesítve a fürtmérettel. A vonal egyik ábrán sem egyenes, ez azt jelenti, hogy a méretezhetőség nem lineáris, azaz nem a leginkább megfelelő.

Ha összegyűjtjük az Apache 1.3.14 és a 2.08a méretezhetőségi adatait (*14. kép*) és grafikonná alakítjuk (*15. kép*), akkor azt figyelhetjük meg, hogy egymáshoz képest mindkét kiszolgáló azonos méretezhetőséggel büszkélkedhet.

Linux-rendszerekben a kiszolgáló mindkét változata egyforma méretezhetőségi adatokkal rendelkezik. Eredményeink alapján az Apache 2.08a két százalékkal jobban méretezhető, mint az 1.3.14-es változat. Mindkét esetben lassú lineáris csökkenést tapasztalhatunk. Nyolc gépnél több elemből álló rendszereknél minél több gépet helyezünk a fürtbe, gépenként annál kisebb teljesítményt kapunk. Nézzük a Java-alapú kiszolgálókat! Bár a Tomcat jobb teljesítményt ért el (több kérelmet dolgozott fel másodpercenként), mint a Jigsaw, ennek ellenére bizonyos méretezhetőségi gondok merültek fel vele kapcsolatban. A *16. képen* az látszik, hogyha több gépet helyezünk el a fürtben, akkor az egyes gépek teljesítménye csökken. Azonban ezt a méretezhetőségcsökkenést számos okkal magyarázhatjuk.

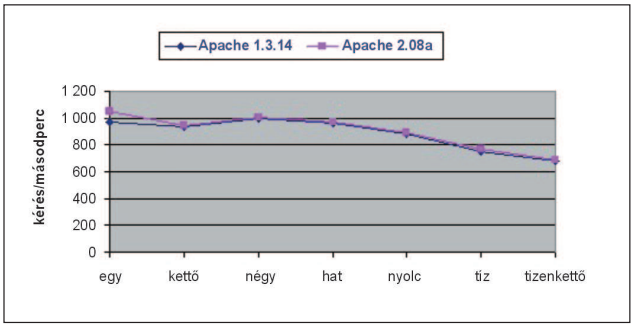
	1.1.14	2.08a
Egy	971	1053
Kettő	937	945
Négy	1000	1003
Hat	964	974
Nyolc	886	892
Tíz	750	764
Tizenkettő	683	685
Átlag	884	902

14. kép A méretezhetőségi adatok összehasonlítása

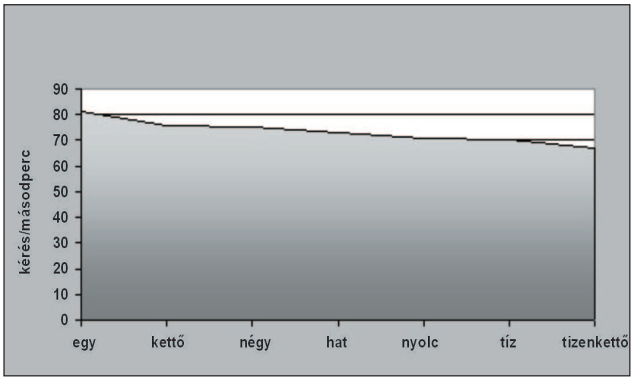
Az eredményeket befolyásoló tényezők

A próbák eredményeit az alábbi tényezők befolyásolhatták:

1. A WebBench próbacsomagját egy NFS-kiszolgálón tettük elérhetővé az ügyfélgépek számára. Így az NFS-nél is jelentkezhet dugulás, amikor másodpercenként több száz ügyfél próbálja meg elérni az NFS-ben tárolt fájlokat.
2. A Jigsaw és a Tomcat Java-alapú webkiszolgáló, így teljesítményük nagymértékben függ Java Virtuális Gép teljesítményétől, melyet egyébként szintén egy NFS-lemezrészről indítottunk el (hiszen a gépek nem tartalmaznak merevlemezt, és a tárhelyet az NFS segítségével osztják meg egymás között).
3. A forgalom gerjesztését csupán 16 Celeron-géppel végezhettük. Az így létrejött forgalom nem biztos, hogy elég volt a gépek túlterheléséhez, és különösen igaz ez az Apache esetében, ahol hatnál több gépet is kipróbáltunk.



15. kép Az Apache 1.3.14 és 2.08a méretezhetőségének összehasonlítása

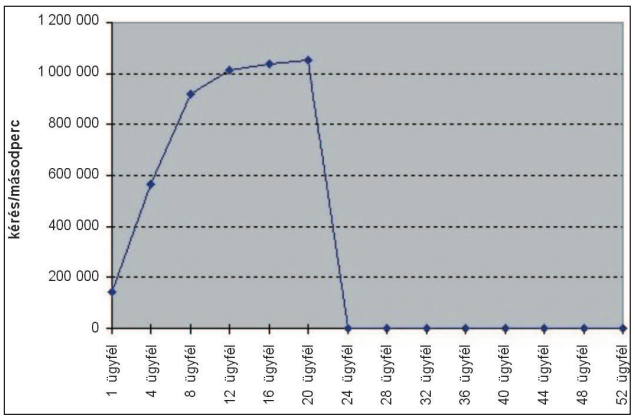


16. kép A Tomcat méretezhetőségi táblázata

A gondok

Munkánk során számos gonddal szembesültünk: nem megfelelő gépek, kísérleti jellegű alkatrészek és programok, nem támogatott meghajtók és eszközök. Ebben a szakaszban csak azokat a gondokat sorolom fel, melyekkel a próbák végrehajtásához szükséges munkák során találkoztunk.

A ZNYX ethernet linuxos meghajtókkal gondjaink voltak a megbízhatóság tekintetében. Ezek a meghajtók még mindig fejlesztés alatt állnak, a kipróbálás idején még nem voltak piacra dobható állapotban. Ha az egy másodpercre jutó adatátviteli műveletek viszonylag magas számot értek el, a meghajtó egyszerűen lefagyott. Nézzünk például egy próbát, melyet egy Apache 2.08a-t futtató gépen végeztünk el! Amikor a gép terhelése eléri a másodpercenként 1053 kérelemkiszolgálást (az átvitt adatmennyiség 6 044 916 bájt volt másodpercenként), az ethernetmeghajtó lefagyott és a ZNYX-csatlakozókkal elvesztettük a kapcsolatot (*17. kép*).



17. kép A túlterheléstől lefagyott ethernetmeghajtó

A ZNYX munkatársai segítségével számos próbát és hibakeresést végrehajtottunk, végül sikerült kiigazítanunk a meghajtó hibáját és a továbbiakban ezen a téren semmiféle gond nem merült fel.

A fürt indításakor a megoldásra váró második gond az `inetd`-vel kapcsolatos gond volt. Az `inetd` démon a többi rendszerszolgáltatás vezérlőjeként működik. A háttérben maradva figyel a hálózati kapukon érkező kapcsolati kérelmekre. Ha létrejön egy kapcsolat, az `inetd` a megfelelő kapuhoz tartozó szolgáltatás démonjának egy példányát indítja el. A gond a mi esetünkben az volt, hogy az `inetd` ismeretlen okból megtagadta az UDP-kérelmek kiszolgálását és minden ilyen esetben újra kellett indítanunk a démont. Ez a gond még mindig fennáll, az `xinetd` legfrissebb kiadásának használata sem oldotta meg.

A másik nyilvánvaló nehézség az volt, hogy nem voltunk képesek elegendő forgalmat gerjeszteni a próbagépek túlterheléséhez. Több erőre volt szükségünk, mint amennyivel a próbákat végeztük. Tevékenységünk kezdetekor csak 17 gép állt készen (egy vezérlő- és 16 ügyfélgép) a próbákhoz. Ez lehet az egyik oka annak, hogy miért nem voltunk képesek nagyobb méretezhetőséggel számolni. Most azonban a próbakörnyezetet kibővítettük 63 gépre, s a közeljövőben újrafuttatjuk a próbákat és ismét ellenőrizzük a friss eredményeket.

Összegzés

Az ARIES program azért indult, hogy bebizonyítsuk: lehetséges-e telekommunikációs osztályú internetes kiszolgálókat építeni a Linux és nyílt forrású programok felhasználásával. Több Linux-terjesztéssel, web- és adatfolyam-kiszolgálóval, forgalomelosztó és forgalomkiegyenlítő eljárással, a HA Linux és az egymás kiváltására képes rendszerek működtetését lehetővé tevő terjesztett és naplózó fájlrendszerekkel és megoldásokkal (NFS, ethernet, programból megvalósított RAID) kísérleteztünk. A jövőben az ARIES-ben végzett munka arra fog irányulni, hogy a Linux méretezhetőségi képességeit növeljük, hogy a rendszer ezáltal a már felállított webkiszolgáló-alkalmazások mellett más mobil internetes szolgáltatásokat is képes legyen futtatni. A fő cél az lesz, hogy a rendszer a fürt erőforrásait minél hatékonyabban használja ki, illetve hogy a mobil internetes alkalmazásokban oly fontos szerepet játszó biztonság is magasabb színvonalra emelkedjen. Továbbá a projekt a felépített rendszerek képességeinek körét az IPv6-eljárás bevezetésével fogja tovább bővíteni. A Linux-fürtön jelenleg megtalálható három webkiszolgálót megtartjuk. A kipróbált webkiszolgálók méretezhetősége az egyre több gép beillesztése után nem lineárisan növekedett, azonban igen jó teljesítménnyel és közel lineárisan növekvő méretezhetőséggel rendelkeznek (a próbában legfeljebb 12 gépet használhattunk). Jelenleg a webkiszolgálók legfrissebb változatainak telepítése folyik (Apache 2.0.15a, Jigsaw 2.2.0, Tomcat 3.2). Próbáink eredményei alapján megállapíthatjuk, hogy az Apache jelentősen gyorsabb és megbízhatóbban működik, mint a többi webkiszolgáló. Szeretnénk próbákat végrehajtani a 2.0-s változat végleges kiadásával is. Erre a változatra a fejlesztők ígérete szerint a tökéletesen tiszta kód, jól szervezett I/O-rétegezés és jóval magasabb szintű méretezhetőség lesz jellemző.

Köszönetnyilvánítások

A szerző szeretne köszönetet mondani az Ericsson Research Open Architecture Research Development csapatának a cikk közlésének engedélyezéséért, illetve az Ericsson Research Canada munkatársainak, ***March Chatel***-nek és ***Evangeline Paquin***-nek a próbák során nyújtott segítségéért.

Ibrahim F. Haddad (ibrahim.haddad@ericsson.com) az Ericsson Research montreali központú Open Architecture laboratóriumának dolgozik, kutatási területe a telekommunikációs osztályú kiszolgálóelemek valós idejű vizsgálata teljes IP-környezetben. Jelenleg a Concordia Egyetem számítástudományi tanszékének doktorandusza.

Kapcsolódó címek

Apache HTTP Server Project ➲ http://www.apache.org/httpd.html
Jigsaw ➲ http://www.w3c.org/jigsaw
Java Sun ➲ http://java.sun.com
Linux IPv6 ➲ http://www.bieringer.de/linux/IPv6
Linux Kernel Home Page ➲ http://www.kernel.org
Netcraft ➲ http://www.netcraft.com
Red Hat ➲ http://www.redhat.com
ServerWatch ➲ http://serverwatch.internet.com
Tomcat ➲ http://jakarta.apache.org/tomcat
WebBench ➲ http://www.zdnet.com/etestinglabs/stories/benchmarks/0,8829,2326243,00.html
WWW Consortium ➲ http://www.w3c.org
ZNYX ➲ http://www.znyx.com