Elosztott relációs adatbázis kezelő rendszerek vizsgálata cloud környezetben

Budai Péter István, Dr. Goldschmidt Balázs

Budapesti Műszaki és Gazdaságtudományi Egyetem
Irányítástechnika és Informatika Tanszék
Email: budai@iit.bme.hu, balage@iit.bme.hu

2013. március 6.

Absztrakt

Az IaaS cloud szolgáltatások terjedésével és egyre szélesebb körű elfogadottságával újabb és újabb alkalmazásokat helyeznek át felhő alapú infrastruktúrába, melyek között találunk tudományos célú és hagyományos többrétegű webes alkalmazásokat is. Ezekben közös, hogy általában különálló fizikai vagy virtuális kiszolgálókon elhelyezkedő szoftverkomponensekből állnak, melyek közül szinte mindig találunk valamilyen adatbázist is. Az adatbázist leggyakrabban valamilyen eredetileg nem felhő alapú környezethez kidolgozott, hagyományos vagy elosztott relációs adatbázis kezelő rendszer szolgálja ki.

Kutatásunkban az elosztott relációs adatbázis kezelő rendszereket, azokon belül elsősorban a MySQL Cluster adatbáziskezelőt vizsgáltuk üzemeltetési és teljesítmény szempontok alapján. Előadásunkban bemutatjuk a MySQL Cluster adatbázis kezelő szoftver komponenseit és felépítését, a konfigurációval és az üzemeltetéssel kapcsolatban felmerülő forgatókönyveket, különös tekintettel a cloud környezetben történő alkalmazásra. Ezután összefoglaljuk az adatbázis kezelő teljesítményére vonatkozó tapasztalatainkat, melyeket a klaszter különböző konfigurációi mellett végzett mérések során gyűjtött teljesítmény metrikák alapján állítottunk össze.

1. Bevezetés

A felhő alapú megoldások mára már megkerülhetetlenné váltak mind a hagyományos webes, mind pedig a tudományos alkalmazások világában [1]. A legtöbb alkalmazás mögött valamilyen adatbázist is találunk, melyek változatosak lehetnek az alkalmazott szoftver-megoldás tekintetében.

Célunk az adatbázisok egy meghatározott csoportjának, az elosztott relációs adatbázis kezelő rendszereknek a vizsgálata a felhő alapú környezetben történő alkalmazhatóság és teljesítmény szempontjából.

Cikkünk első részében áttekintjük az elosztott adatbázisok felhő alapú rendszereken való alkalmazásának lehetőségeit, majd részletesebben is bemutatjuk a MySQL Cluster elosztott relációs adatbázis kezelő rendszer tulajdonságait, felépítését és a cloud környezetben való alkalmazása során fellépő forgatókönyveket. A cikket a MySQL Cluster teljesítményét elemző szakasz zárja, mely az adatbázis kezelő skálázódási tulajdonságaira kíván rávilágítani.

2. Elosztott adatbázis kezelő rendszerek a felhőben

Cloud computing-ról, avagy felhő alapú számítástechnikáról akkor beszélünk, amikor a számítási erőforrások, hálózati sávszélesség, tárolási kapacitás és különböző szoftverek szolgáltatásként jelennek meg, amely szolgáltatás távolról, az interneten keresztül vehető igénybe

önkiszolgáló módon és általában a tényleges használat után számlázódik [2].

Jelen kutatásunkban elsősorban a felhő alapú adattárolási megoldásokra koncentrálunk. Az adatok tárolására azok típusától (strukturált rekordok, képek, fájlok) függően számos megoldást kínálnak a népszerű cloud szolgáltatók:

- Nagyméretű bináris objektumokhoz, azaz videó és zenei állományokhoz, tudományos adathalmazokhoz az úgynevezett **blob** (Binary Large OBject) típusú tárolók illeszkednek a legjobban, mint például az Amazon S3 [3] vagy a Windows Azure Blob Storage [4]. Ezek a megoldások leginkább egy hagyományos könyvtár- és állomány-hierarchiához hasonló struktúrát is biztosítanak az adatok csoportosításához.
- Nagyszámú, de egyenként a blob-okhoz képest nagyságrenddel kisebb méretű, strukturált adatok hatékony tárolására a **NoSQL** [5] megközelítést alkalmazó szolgáltatások használhatók, mint például a Google BigTable [6] és az Azure Table Storage [7]. Ezen megoldások a tárolt elemek nagy száma miatt jellemzően csak korlátozott lekérdezési lehetőségeket biztosítanak, például többnyire csak egy kulcs alapján lehet hatékonyan elérni a tárolt adatokat.
- Rekordorientált, erősen strukturált adathalmazok tárolására a hagyományos relációs adatbázis kezelő (**RDBMS**) rendszerek a legalkalmasabbak. Habár a különböző cloud szolgáltatók relációs adatbázisokat is kínálnak önálló szolgáltatásként (ilyen például az Amazon RDS [8] vagy az Azure SQL Storage [9]), gyakori eset, hogy az adatbázisok felhőben történő kialakításához a szintén felhőalapú szolgáltatásként elérhető virtualizált szervereket (VM, VPS) használják fel, melyekre a hagyományos fizikai architektúrákon is alkalmazott adatbázis kiszolgáló rendszereket telepítik. Jelen cikkben is ez utóbbi csoporttal foglalkozunk.

A hagyományos relációs adatbázis kiszolgáló rendszerek felhő alapú infrastruktúrába való telepítésének számos előnye és hátránya van az azonnal igénybe vehető adatbázis szolgáltatásokhoz képest:

- A készen kapható megoldások többnyire kevésbé testre szabhatóak és kevesebb specializált szolgáltatást nyújtanak, ezért nehezebben integrálhatóak egy komplett alkalmazásba. Egy jó példa erre a jogosultságkezelés kérdése, ami a saját virtuális adatbázis kiszolgálók üzemeltetésénél könnyebben kapcsolható hozzá a vállalati authentikációs és authorizációs rendszerhez.
- Egy már meglévő, eddig fizikai szervereken futó alkalmazásnak a felhőbe való áttelepítésekor sokkal gyorsabb eljárás ugyanazt a szoftverkörnyezetet kialakítani a virtuális infrastruktúrán, mint az alkalmazást módosítani, hogy illeszkedjen az új adatbázis kiszolgáló rendszerhez.
- A virtuális infrastruktúrán futó adatbázis szoftverek menedzselésével és karbantartásával kapcsolatos feladatok (monitorozás, szükség esetén kapacitás bővítés, biztonsági mentések készítése és a korábbi állapot helyreállítása) viszont ebben az esetben ugyanúgy ránk hárulnak, mint a hagyományos fizikai környezet esetén. A készen kínált szolgáltatások esetén ezekkel a problémákkal nem szembesülünk.

A cloud környezetben a virtuális kiszolgálók menedzselése sokkal kényelmesebb, mint a fizikai megfelelőiké, ezért a jobb hibatűrési képességekkel és magasabb rendelkezésre állási idővel rendelkező elosztott adatbázis kezelő rendszerek üzemeltetése is gazdaságosan végezhető.

Az elosztott relációs adatbázis kezelő rendszerek között megkülönböztetünk **homogén** és **heterogén** rendszereket. Az előbbi esetben minden kiszolgálón ugyanolyan szoftverkörnyezet található (az adatbázis kiszolgáló szoftver és gyakran az operációs rendszer tekintetében is), míg az utóbbi esetben vegyes komponensekből áll össze a teljes rendszer. Az elosztottság szempontjából beszélhetünk **redundanciá**ról vagy **replikáció**ról, amikor az egyes adatrekordok több kiszolgálón is megtalálhatóak a jobb hibatűrés érdekében, illetve **partícionálás**ról, amikor az

adatok az egyes kiszolgálókon nem többszörözve, hanem elosztva találhatók meg, így növelve a tárolási kapacitást és a feldolgozási sebességet. Ezek két tulajdonság kombinálásával skálázható, hibatűrő adatbázis rendszereket alakíthatunk ki.

Kutatásunkban lehetőleg nyílt forráskódú vagy ingyenesen hozzáférhető elosztott relációs adatbázis kezelő rendszereket vizsgáltunk a felhő alapú környezetben való alkalmazhatóság és a skálázódással kapcsolatos viselkedésük szempontjából.

3. MySQL Cluster

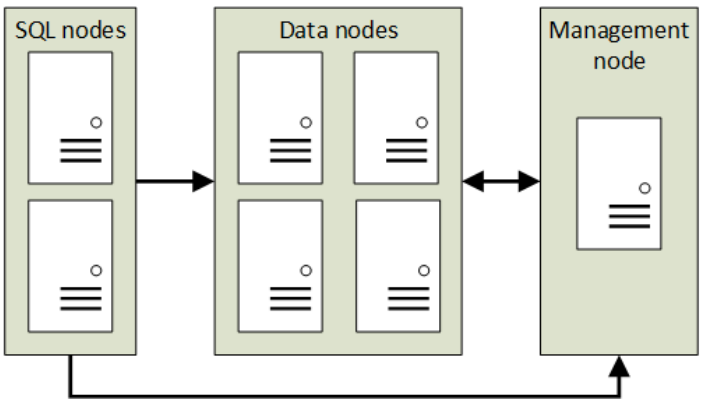
A MySQL Cluster egy nyílt forráskódú elosztott relációs adatbázis kezelő rendszer, melyet a népszerű MySQL adatbázis kezelő egy különálló ágaként fejlesztenek. A fejlesztés eredménye egy homogén, fizikailag teljesen elosztott relációs adatbázis, mely magas rendelkezésre állást és redundanciát kínál [10].

A MySQL Cluster legmeghatározóbb jellemzője, hogy az adatokat teljes mértékben a kiszolgáló memóriájában tárolja, és csak a tranzakciós naplókat írja folytonosan a merevlemezre. Természetesen a kiszolgáló leállításakor nem vesznek el az adatok, olyankor a teljes adatbázis a merevlemezre íródik.

A teljesen memóriaorientált működés előnye egyrészt a nagyobb teljesítményben rejlik, a lekérdezések nagyságrendekkel gyorsabban futnak, ha az összes adat azonnal rendelkezésre áll. A másik pozitív hozadék, hogy az elosztott működéshez szükséges szinkronizációs algoritmusok tovább optimalizálhatók, ha nem kell a háttértáron lévő adatok konzisztenciájáról is gondoskodni. A megoldás hátránya, hogy a tárolható adatmennyiség a memória korlátozott kapacitása miatt messze elmarad a hagyományos adatbázisokhoz képest. Ezt felismerve, az újabb verziókban már lehetőség van a nem indexált adatok háttértárra történő tárolására is.

A MySQL Cluster adatbázis kezelő rendszert úgy tervezték, hogy egyik elem meghibásodása se okozza a szolgáltatás kiesését. Egy MySQL Cluster konfiguráció alapvetően háromféle szoftverkomponensből áll, melyek mindegyike különálló kiszolgálón kerül telepítésre, ahogy az 1. ábrán is látható:

- **SQL node**: Az SQL kiszolgálók a hagyományos relációs adatbázis kezelő rendszerekben megszokott módon viselkednek, feladatuk a kliensek felől érkező kapcsolatok fogadása, az SQL lekérdezések értelmezése és a feldolgozási terv előkészítése, melyet azután az adatkiszolgálók felé továbbítanak.
  A gyakorlatban egy hagyományos MySQL kiszolgálóként látszanak, melyekben a relációs táblák a lokális MyISAM vagy InnoDB adatbázis motorok helyett az elosztott NDBCluster motort használják.
  A hibatűrő működés eléréséhez, illetve a lekérdezések további párhuzamosítása érdekében legalább két SQL kiszolgáló telepítése ajánlott.
- **Data node**: Az adat kiszolgálók feladata az adatok tényleges tárolása az egyes kiszolgálók között elosztva, többszörözve és szinkronizálva.



1. ábra: A MySQL Cluster felépítése

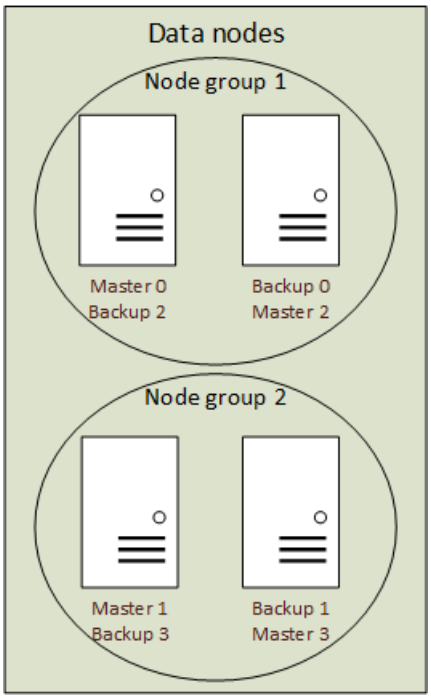
Egy MySQL Cluster konfiguráció összeállítása során az előírt hibatűrési követelmények alapján választhatjuk meg, hogy milyen mértékű redundanciát és mennyi adatkiszolgálót alkalmazunk. Ezen változók alapján a menedzsment kiszolgáló úgy-nevezett kiszolgáló csoportokba (node group) szervezi az adatkiszolgálókat. Egy csoport annyi elemű, amilyen szintű redundanciát alkalmazunk. Az adatkiszolgálók száma csak a redundancia érték többszöröse lehet, és ez fogja meghatározni a csoportok számát a klaszterben.

Az adatkiszolgálókkal párhuzamosan maguk a tárolt adatok is partícionálásra kerülnek. Minden csoportba az adatok két különböző partíciója kerül, melyek többszörözve, a csoportba tartozó minden kiszolgálón tárolásra kerülnek, ezt mutatja be a 2. ábra.

Ennek a felépítésnek köszönhetően az adatok mindaddig elérhetőek maradnak, amíg mindegyik csoportból legalább egy kiszolgáló működőképes, amit megfelelő redundancia érték választással biztosíthatunk. Az adatkiszolgálók száma leginkább az adatbázis kapacitására van hatással (futás közben is van lehetőség a klaszter új csoporttal való bővítésére), de előfordulhat olyan alkalmazási terület, ahol az adatok szempontjából erősen lokalizált lekérdezések történnek, így nagyobb fokú párhuzamosítás is elérhető a partíciószámok növelésével.

- **Management node**: A menedzsment kiszolgáló feladata a klaszter felépítése és a többi komponens monitorozása. A klaszter indításakor az egyes elemek a menedzsment kiszolgálóra való kapcsolódással értesülnek az adatbázis kezdeti konfigurációjáról és ezen keresztül találják meg egymást.

  A menedzsment kiszolgáló az adatok kezelésében közvetlenül nem vesz részt, csak a klaszter konfigurációjának változásakor (hálózati vagy adatbázis beállítások változása, a klaszter bővítése új csoportttal) van szerepe, így a kiesése nem befolyásolja az adatbázis elérhetőségét. Emiatt ezt a komponenst nem szükséges duplikálni, bár a lehetőség meg van rá.

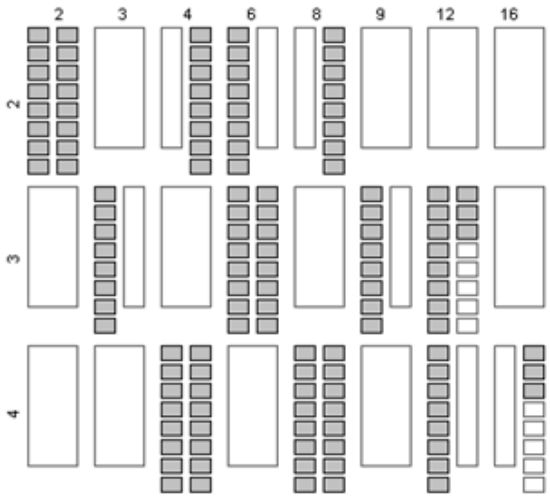


2. ábra: Adatok partícionálása a MySQL Cluster rendszerben

Egy MySQL Cluster adatbázis üzemeltetése során első lépésként fel kell térképezni, hogy az adatbázist majd később felhasználó alkalmazás milyen fokú hibatűrést igényel és nagyságrendileg mekkora adatmennyiséggel kell számolni. Ezen információk alapján kiválasztható a redundancia mértéke, és a szükséges SQL és adatkiszolgálók száma.

A hatékony működéshez a MySQL Cluster a komponensei között. egy nagysebességű, dedikált TCP/IP hálózat megléte ajánlott, aminek teljesítéséhez a felhőben történő üzemeltetés esetén a cloud szolgáltató támogatása is szükséges. Ha ez nem valósítható meg, akkor törekedni kell a kiszolgálók egy rendelkezésre állási zónába való telepítésére.

A már üzembe helyezett MySQL Cluster rendszer alapvetően nem igényel különleges karbantartási feladatokat, egyedül az adatbázis méretét kell gyakrabban monitorozni a memória szűkösebb tárolókapacitása miatt.



3. ábra: A vizsgált konfigurációk (a vízszintes osztásokban az adat és SQL kiszolgálók száma, függőlegesen pedig a redundancia és a lekérdezés típusok)

4. MySQL Cluster teljesítményvizsgálata

A MySQL Cluster vizsgálatakor elsősorban arra a kérdésre kerestük a választ, hogy annak teljesítménye hogyan változik, ha különböző dimenziók (redundancia, klasztermeret, adatbázis méret) mentén skálázni kezdjük az adatbázis kezelő rendszert.

A méréshez összeállítottunk egy minta adatbázis sémát, melyet úgy alakítottunk ki, hogy különböző típusú lekérdezések is végrehajthatóak legyenek rajta, többek között indexelt és nem indexelt oszlopra történő keresés, kettő és több relációs tábla illesztése (JOIN), sorokat aggregáló (SUM, AVG) és csoportosító (GROUP BY) műveletek. Ezek közül a vizsgálatba 7 jellemző lekérdezés fajtát választottunk ki.

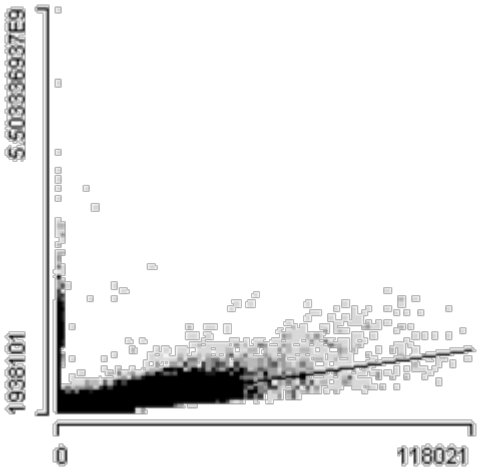
A vizsgálat során a MySQL Cluster 7.2 rendszer 17 különböző konfigurációját állítottuk össze az Amazon EC2 cloud infrastruktúrában elérhető **medium** típusú kiszolgálókból, melyek egymástól a beállított redundancia mértékében, illetve az adat és SQL kiszolgálók számában tértek el. A legkisebb mérési konfiguráció így 4, míg a legnagyobb 19 virtuális gépből állt. A megvizsgált konfigurációkat a 3. ábrán foglaltam össze.

A mérések során egy több kliens virtuális gépből álló automatikus rendszer hajtotta végre a lekérdezéseket és mérte egyenként azok lefutási idejét. Annak érdekében, hogy az esetleges kiugró vagy hibás mérési adatokat minimalizáljuk, a lekérdezéseket nagy ismétlésszámmal hajtottuk végre, méghozzá úgy, hogy mindegyik kliens felváltva szólította meg a SQL kiszolgálókat és véletlenszerűen változtatták a lekérdezések paramétereit is, hogy az esetlegesen jelen lévő gyorsítótárak (cache) hatásait minimalizáljuk.

A méréseket minden konfigurációban különböző adatbázis méretek mellett is megismételtük, így végül több, mint 350 000 egyedi lekérdezésről rendelkeztünk adatokkal, melyeket ezután elemzés alá vettünk.

A lefutási idők általános eloszlását megfigyelve azt tapasztaltuk, hogy a lekérdezések döntő többsége, mintegy 99 százaléka két másodpercen belül lefutott (volt olyan lekérdezés, aminek végrehajtásához alig több, mint 1,93 milliszekundumra volt szükség), míg a fennmaradó 1 százalék változatos eredményeket hozott egészen a 70,6 másodperces maximális értékig.

Először ezeket a kiugró, az eredményeket torzító értékeket eltávolítva próbáltunk meg összefüggést felfedezni a lekérdezések futásideje és a klaszter konfigurációja között, de nem jártunk sikerrel. Nem tapasztaltunk szignifikáns korrelációt sem a redundancia értékével, sem a partíciók számával, sem az adatbázis méretével. Egyedül a lekérdezés eredményhalmazának mérete, az abban szereplő adatbázis sorok száma mutatott összefüggést a lefutási idővel (4. ábra), mely visszavezethető a hálózati átviteli időre.



4. ábra: A lefutási idők (függőleges) és az eredményhalmaz méretének (vízszintes) kapcsolata

5. ábra: Balra fent az adatbázis méret (függőleges) és a szűrőtartomány (vízszintes), jobbra fent az adatbázis méret és a lefutási idők (vízszintes), jobbra lent pedig a szűrőtartomány (függőleges) és a lefutási idők közti kapcsolat (a színes pontok összetartozó adatpontokat jelölnek)

A következőkben az előző elemzésben kihagyott kiugró értékekre koncentráltunk. Megfigyeltük, hogy a kiugró értékek mind ugyanahhoz a lekérdezés típushoz tartozó lefutási időket takarják. Az érintett lekérdezés egy összetett, táblák illesztésével, átlagszámítással és nem indexelt oszlopra való tól-ig típusú szűréssel is járó kifejezés volt. A további vizsgálataink a lefutási idő kapcsán külön-külön összefüggést mutattak mind az adatbázis méretével, mind pedig a lekérdezésben szereplő szűrőtartomány méretével, ám míg az előbbivel pozitív, addig az utóbbi esetén negatív korrelációval.

A kapott eredmények alapján elmondható, hogy a MySQL Cluster adatbázis kezelő rendszer teljesítményben nem érzékelhető módon skálázódik a redundancia, a kiszolgálók száma és az adatbázis méret tekintetében is, viszont bizonyos típusú lekérdezések (nem indexelt oszlopra való szűrés) kiszolgálásakor érzékennyé válik a paraméterekre és az adatbázis méretre.

5. Kapcsolódó munkák

A MySQL Cluster rendszer teljesítményét már számos esetben vizsgálták, többek között maga a gyártó Oracle vállalat is [11]. Az általunk felhő alapú infrastruktúrán elvégzett méréssel ellentétben, ott dedikált hardverkomponenseken végezték a vizsgálatot. Továbbá az abban a cikkben szereplő eredményeket sem sikerült reprodukálnunk, mi nem tapasztaltunk teljesítménybeli javulást a kiszolgálók számának növekedésével.

A megállapításaink összhangban vannak a MySQL Cluster hivatalos telepítési útmutatójában [12] szereplő kijelentéssel, mely szerint az adatbázis kiszolgáló rendszert úgy optimalizálták, hogy elsősorban az elsődleges kulcs szerinti keresések legyenek hatékonyak.

6. Összefoglalás

Kutatásunkban a felhő alapú szolgáltatásokon elérhető adattárolási lehetőségeket vizsgáltuk, különös tekintettel az elosztott relációs adatbázis kezelő rendszerek cloud környezetben történő alkalmazására és teljesítményére.

Alapos vizsgálatnak vetettük alá a MySQL Cluster adatbázist, tanulmányoztuk és bemutattuk a felépítését, majd skálázódási vizsgálatoknak vetettük alá felhő alapú infrastruktúrán.

A kutatásunk folytatásaként újabb elosztott relációs adatbázis kezelő rendszereket tervezünk megvizsgálni a MySQL Cluster vizsgálata során kidolgozott teszt keretrendszer felhasználásával.

7. Köszönetnyilvánítás

A jelen cikk alapjául szolgáló munka szakmai tartalma kapcsolódik a "Új tehetséggondozó programok és kutatások a Műegyetem tudományos műhelyeiben" c. projekt szakmai célkitűzéseinek megvalósításához. A projekt megvalósítását a TÁMOP - 4.2.2.B-10/1--2010-0009 program támogatja.

8. Hivatkozások

[1] E. Deelman, G. Singh, et. al: *The Cost of Doing Science on the Cloud: The Montage Example*, Proceedings of the ACM/IEEE conference on Supercomputing (SC '08), Article No. 50, 2008

[2] R. Buyya, J. Broberg, A. Goscinski: *Cloud Computing: Principles and Paradigms*, Wiley, 2011, ISBN 978-0-470-88799-8

[3] *Amazon Simple Storage Service (Amazon S3)*, http://aws.amazon.com/s3/, letöltve: 2013. 03. 06.

[4] *How to use the Windows Azure Blob Storage Service in .NET*, http://www.windowsazure.com/en-us/develop/net/how-to-guides/blob-storage/, letöltve: 2013. 03. 06.

[5] R. P. Padhy, M. R. Patra, and S. C. Satapathy: RDBMS to NoSQL: *Reviewing Some Next-Generation Non-Relational Databases*, International Journal of Advanced Engineering Science and Technologies 11, no. 1, p. 15-30, 2011

[6] F. Chang, J. Dean, et. al: *Bigtable: A Distributed Storage System for Structured Data*, Seventh Symposium on Operating System Design and Implementation (OSDI'06), Seattle, November 2006.

[7] *How to use the Table Storage Service*, http://www.windowsazure.com/en-us/develop/net/how-to-guides/table-services/, letöltve: 2013. 03. 06.

[8] *Amazon Relational Database Service (Amazon RDS)*, http://aws.amazon.com/rds/, letöltve: 2013. 03. 06.

[9] *How to use Windows Azure SQL Database in .NET applications*, http://www.windowsazure.com/en-us/develop/net/how-to-guides/sql-database/, letöltve: 2013. 03. 06.

[10] *MySQL Cluster referencia kézikönyv*, http://dev.mysql.com/doc/refman/5.5/en/mysmy-cluster.html, letöltve: 2013. 03. 04.

[11] M. Keep: *Performance Testing of MySQL Cluster: The flexAsynch Benchmark*, https://blogs.oracle.com/MySQL/entry/performance_testing_of_mysql_cluster, letöltve: 2013. 03. 06.

[12] *MySQL Cluster Evaluation Guide*, February 2013.