DEBRECENI EGYETEM
INFORMATIKAI KAR
INFORMATIKAI RENDSZEREK ÉS HÁLÓZATOK TANSZÉK

Elosztott adattárolás és magas rendelkezésre állás megvalósítási lehetőségei linux operációs rendszer alatt

Témavezető:
Dr Ecsedi Kornél
ügyvivő-szakértő, csoportvezető

Készítette:
Egri Zsolt
Programtervező Informatikus

Debrecen
2009

Tartalomjegyzék

1. Bevezetés ... 3

1.1. Köszönetnyilvánítás ... 3

1.2. Célkitűzések ... 4

2. Elosztott adattárolás fogalmi köre ... 5

2.1. Osztott fájlrendszer ... 5

2.2. Megosztott tárolóeszköz alapú fájlrendszerek ... 6

2.3. Elosztott fájlrendszerek jellemzői ... 6

2.3.1. Authentikáció ... 6

2.3.2. Zárkezelés ... 7

2.3.3. Teljesítmény ... 8

2.3.4. Hibatűrés ... 8

2.3.5. Gyorsítótár ... 9

3. Tesztkörnyezet ... 9

3.1. A coLinux telepítése ... 10

3.1.1. Hálózat coLinux alatt ... 10

3.2. A vmware player telepítése ... 11

3.3. Teszt operációs rendszer ... 11

3.3.1. Hálózati beállítások ... 11

3.4. Teljesítménymérés, benchmarking ... 12

3.4.1. Egyszerű írási sebesség mérés ... 12

3.4.2. Postmark ... 13

3.4.3. IOZone ... 14

4. Gyakorlati megvalósítások ... 15

4.1. Hálózati fájlrendszerek ... 15

4.1.1. NFS ... 16

4.1.2. Samba ... 20

4.2. Elosztott fájlrendszerek ... 23

4.2.1. OpenAFS ... 24

4.3. Párhuzamos működésű hibatűrő elosztott fájlrendszerek ... 26

4.3.1. GlusterFS ... 27

4.4. Osztott tárolóeszköz alapú filerendszerek 32
4.4.1. GFS (Global FileSystem) 33
4.5. Peer-To-Peer fájlrendszerek 34
5. Adatbázis elosztási lehetőségek 35
5.1. Replikációk 35
5.1.1. MySQL replikáció 36
5.2. Adatbázis klaszterek 39
5.2.1. MySQL klaszter 40
6. Magas rendelkezésre állás biztosítása 46
6.1. Heartbeat, Pacemaker 46
7. Összefoglalás 51
8. Irodalomjegyzék 52

1. Bevezetés

Napjainkban az informatika rohamos léptekkel fejlődik. A fejlődés velejárójaként született meg az igény a minél jobban skálázhatóságra, illetve minél nagyobb rendelkezésre állásra. A hardver elemek egyre olcsóbbá, egyre nagyobb teljesítményűvé válnak, a szoftverek teljesítmény igénye viszont még ennél is gyorsabban növekszik. Bármely szakterületre is tekintünk hamar meglátjuk a jelentőségét e két kulcstényezőnek, melyek rendszereink minőségén javíthatnak. Ma már nem elég létrehozni egy rendszert, amely működik. Olyan rendszerekre van szükségünk, melyek könnyedén bővíthetőek, nem igényelnek strukturális változtatást a bővítéshez és mindemellett meg van a képességük a működésre bizonyos hibák bekövetkeztekor is, tehát hibatűrőek. Dolgozatom célja feltérképezni a szóba jöhető szabad szoftveres megoldásokat és rávilágítani, mely megoldások milyen előnyökkel és hátrányokkal használhatóak, illetve bemutatni működés közben egyes lehetőségeket.

1.1. Köszönetnyilvánítás

Elsőként szeretném megköszönni mindazoknak, akik idejükkel, szaktudásukkal vagy anyagilag hozzájárultak az open source közösségek létrejöttéhez, és lehetővé tették, hogy a dolgozatban vizsgált programok és rendszerek létrejöhessenek. Nagy segítséget jelentett dolgozatom megszületéséhez a közösségi szerkesztés alatt álló és folyamatosan bővülő Wikipedia Internetes enciklopédia rendszer illetve a szabad forrású szoftverek dokumentációi, és online leírásai. Szeretném végül, de nem utolsósorban megköszönni a segítséget témavezetőmnek, Dr. Ecsedi Kornélnak, aki észrevételeivel és tapasztalataival nagyban hozzájárul dolgozatom színvonalának emeléséhez.

1.2. Célkitűzések

Dolgozatomban nem kívánom lefedni a teljes témakört, mely az elosztott rendszereket érinti, mert a témakör terjedelme túlmutat jelen dolgozat keretein. Célom, hogy átfogó képet nyújtsak az elosztott tárhely és adatbázis lehetőségekről, azok nyílt forrású megjelenéseiről illetve bepillantást adjak a magas rendelkezésre állású rendszerek megvalósításába.

Dolgozatomban szeretném bemutatni, hogyan érhető el nyílt forrású és ingyenes alkalmazások felhasználásával az elosztott adattárolás és ehhez kapcsolódóan a magas rendelkezésre állás. Rövid fogalmi összefoglalást követően sorra veszem azon fájlrendszereket, melyek ezen adatelosztási feladatban megoldást nyújthatnak, majd pedig bemutatom, milyen lehetőségek találhatóak az adatbázis rétegben. A cél egy transzparens rendszer létrehozása, mely anélkül biztosít magas rendelkezésre állást, és elosztott adattárolást, hogy ehhez a felhasználói programok szintjén módosításokat kellene eszközölnünk, tehát felhasználói szemmel nézve transzparens.

2. Elosztott adattárolás fogalmi köre

Az általánosan elfogadott meghatározás alapján "Az elosztott rendszereket olyan önálló, hálózaton keresztül összekapcsolt számítógépek alkotják, amelyek közösen, egymással együttműködve képesek megoldani egy adott feladatot, vagy lehetővé teszik több, egymással valamilyen logikai kapcsolatban álló program párhuzamos végrehajtását" (*Crichlow* [2003] 2.o.). A fenti fogalmat Crichlow azzal bővíti, hogy a felhasználó szempontjából a rendszer legyen transzparens.

Dolgozatomban főként az elosztott adattárolásról, még közelebbről pedig az elosztott fájlrendszerekről, illetve adatbázis replikációkról és adatbázis klaszterekről lesz szó, ezért most részletesebben ezek fogalmait ismertetem röviden.

2.1. Osztott fájlrendszer

A wikipedia[1] szerint a számítástechnikában, egy osztott fájlrendszer úgy teszi lehetővé a fájlokhoz való hozzáférést egy másik távoli gépen, mintha a saját számítógépünkön dolgoznánk. Ez lehetővé teszi, több felhasználó számára, több gépről fájlok és egyéb erőforrások megosztását.

A kliens csomópontok nem rendelkeznek közvetlen hozzáféréssel a mögöttes blokk tároló eszközhöz, hanem a hálózaton keresztül egy protokollt használva kommunikálnak. Ez lehetővé teszi, a hozzáférés korlátozását hozzáférési listák segítségével a szerver vagy a kliens oldalon, attól függően, hogy milyen protokollt használunk. Az osztott fájlrendszerekre a dolgozatban **távoli fájlelérés** néven is hivatkozok.

[1] A következő internetes cím alapján, saját fordítás: http://en.wikipedia.org/wiki/Distributed_file_system, letöltve: 2009.10.26.

Osztott fájlrendszerek képesek lehetnek replikációk és hibatűrő mechanizmusok kezelésére. Vagyis, ha egy korlátozott számú csomópont egy fájlrendszerben offline, a rendszer továbbra is működik anélkül, hogy adatvesztés történne.

2.2. Megosztott tárolóeszköz alapú fájlrendszerek

Az osztott fájlrendszerekkel szemben a **megosztott tárolóeszköz alapú fájlrendszerek** minden csomópont egyenlő mértékben férhetnek hozzá a blokk tároló eszközhöz, ahol a fájlrendszer található. Ezeken a rendszereken hozzáférés-ellenőrzést a kliens oldalon kell megvalósítani.

Megosztott tárolóeszközre alapuló fájlrendszer esetében általában nem biztosított a hibatűrő működés, legalábbis önmagában a fájlrendszer ehhez nem biztosít eszközöket. Ezen esetekben általában külön megoldást kell keresnünk a tárolóeszköz rendelkezésre állásának növelésére.

2.3. Elosztott fájlrendszerek jellemzői

A gyakorlati részben tárgyalt példákban a következőkben bemutatásra kerülő jellemzőket fogom vizsgálni.

2.3.1. Authentikáció

Az authentikáció a görög αυθεντικός: 'valódi', 'eredeti' jelentésű szóból származik, és az informatikában olyan eljárást jelent, amelynek során meggyőződünk arról, hogy valamely információ (legtöbbször egy felhasználó azonosságára vonatkozó állítás) megfelel a valóságnak. Legjobb magyar megfelelője a "hitelesítés" szó lehet. Az informatikai rendszerek különböző fokú (megbízhatóságú) hitelesítési eljárás után adnak jogosultságot erőforrásaikhoz.[2]

[2] A következő internetes cím alapján: http://fogalomtar.eski.hu/index.php/Authentikáció, letöltve: 2009.10.11.

A vizsgált megoldásoknál két szemszögből is körbejárjuk az authentikáció kérdését. Egyrészt megvizsgáljuk, hogy milyen lehetőségeket biztosít az adott fájlrendszer a csatlakozó csomópontok hitelesítésére, másrészt pedig, hogy hogyan illeszkedik az operációs rendszer authenikációs sémájába.

Azon esetekben, ahol nem megfelelő az authentikáció, azt általában könnyen orvosolhatjuk valamilyen VPN megoldás segítségével. Ez viszont a teljesítmény rovására mehet.

2.3.2. Zárkezelés[3]

A fájl zárolás azon mechanizmus, mely hozzáférési megszorításokat biztosít egy számítógépes fájlra azáltal, hogy csak egy felhasználónak, vagy processzusnak biztosít hozzáférést egy adott időpillanatban. A fájlzárolás célja a klasszikus konkurens frissítés esetének elkerülésére.

A konkurens frissítések problémája a következő példával szemléltethető:

1. A processzus beolvas egy rekordot egy fájlból, mely számla információkat tartalmaz a vásárló számla egyenlegével és telefonszámával.
2. B processzus ugyanazon rekordot olvassa be ugyanazon fájlból, így van egy saját másolata.
3. A processzus megváltoztatja az egyenleget a saját vásárló rekord másolatában, és visszaírja a rekordot a fájlba.
4. B processzus - melynek még mindig az eredeti, elévült értéke van meg a saját vásárlói adat rekordjában - frissíti a vásárló telefonszámát, és visszaírja a vásárló adatait a fájlba.
5. B processzus ezzel felülírta az elévült egyenlegével a rekordot a fájlban, és ezzel az A processzus által véghezvitt módosításokat elveszítettük.

[3] A következő internetes cím alapján, saját fordítás: http://en.wikipedia.org/wiki/File_locking, letöltve: 2009.10.18.

A fájl zárolás megelőzi a problémát azzal, hogy sorosításra kényszeríti a frissítő processzusokat minden fájl esetében. A legtöbb operációs rendszer támogatja a rekord zárolás koncepcióját is, mely azt jelenti hogy független rekordok egymástól függetlenül zárolhatóak bármely fájlban, ezzel megnövelve a konkurens frissítő processzusok számát.

2.3.3. Teljesítmény

A teljesítmény igen fontos jellemzője egy fájlrendszernek. Így van ez az elosztott adattárolásban is. A cél, hogy a rendelkezésre álló erőforrásokat minél jobban kihasználjuk, minél nagyobb teljesítményt érjünk el adott hardver felhasználása mellet. Gyakori mérőszámok az adatátviteli sebesség (írás, olvasás, újraítás, stb.) illetve a könyvtárműveletek (létrehozás, törlés, átnevezés, mozgatás, stb.) A teljesítmény mérésével kapcsolatosan bővebben szót ejtek a dolgozat gyakorlati részében.

2.3.4. Hibatűrés

A hibatűrés bizonyos modern fájlrendszerek esetében magában a fájlrendszerben kerül megvalósításra. Az alap elgondolás az, hogy amennyiben egy vagy több fájlszerver meghibásodik, ez ne jelentse a fájlszolgáltatások leállását. Ezt általában valamilyen tükrözési módszerrel érik el a fájlrendszerek. Amennyiben a fájlrendszer nem nyújt megoldást a hibatűrő működésre, igen nehéz azt más módszerekkel biztosítani, mert a fájlrendszer integritása nehezen garantálható a fájlrendszer bevonása nélkül.

2.3.5. Gyorsítótár

A gyorsítótár vagy cache (ejtsd: „kes”) francia eredetű kifejezés (jelentése: „rejtekhely”, „rejtett”), a számítástechnikában az átmeneti információtároló elemeket jelenti, melyek célja az információ-hozzáférés gyorsítása. A gyorsítás egyszerűen azon alapul, hogy a gyorsítótár gyorsabb tárolóelem, mint a hozzá kapcsolt, gyorsítandó működésű elemek, így ha ezen területek tartalma korábban már bekerült a gyorsítótárba (mert már valaki/valami hivatkozott rá korábban), az ilyen adatokat nem a lassú működésű területről, hanem a gyors cache tárolóból lehet előhívni.[4]

A gyorsítótár kezelésnek nagy jelentősége van az osztott fájlrendszerek teljesítményében, mert a gyorsítótárból kiszolgált elemek, legyen az fájl vagy memória alapú gyorsítótár, jóval gyorsabb hozzáférést biztosít az adatokhoz, mint a hálózaton keresztül történő kommunikáció. Általában a gyorsítótár kezeléshez a fájlrendszer nyújt támogatást, és sok esetben megadható, hogy milyen mértékű vagy milyen technikájú gyorsítótár kezelést szeretnénk alkalmazni. A nehézséget általában a gyorsítótár invalidálása, vagy visszavonása jelenti.

Az invalidálás az a folyamat, melynek során a fájlrendszer gyorsítótárában szereplő adatok elavulnak és eltávolításra kerülnek.

3. Tesztkörnyezet

A tesztkörnyezet kialakítására két lehetőséget vizsgáltam meg. Az egyik a coLinux nevű nyílt forrású szoftver volt, mely gyorsabb működést, és könnyebb integrációt ígért a meglevő windows 7 operációs rendszeremhez. Sajnos a coLinux jelenleg nem támogatja a 64 bites operációs rendszereket, így ezen a vonalon elakadtam. A másik lehetőség a vmware nevű szoftver jelenti. Ez a kitűnő ingyenes szoftver végül minden szükséges elvárásnak eleget tett. A következő fejezetekben röviden bemutatom mindkét lehetőség telepítését, és szót ejtek néhány fontosabb jellemzőjükről.

[4] A következő internetes cím alapján: http://hu.wikipedia.org/wiki/Gyorstár, letöltve: 2009.10.18.

3.1. A coLinux telepítése

A különböző elosztott rendszerek teszteléséhez megpróbálkoztam a coLinux nevű szabad forrású linux emulátor programmal is. Azért gondoltam ezt jó választásnak, mert lehetővé teszi, hogy az operációs rendszer újratelepítése nélkül próbáljuk ki az egyes programokban rejlő lehetőségeket, illetve könnyen kialakítható segítségével egyetlen fizikai számítógépen is több virtuális gép, mely elengedhetetlen az elosztott rendszerek képességeinek teszteléséhez. A coLinux viszont sajnos nem támogat bizonyos platformokat, és közel sem olyan rugalmas, mint a teljes számítógépet emuláló társai, így végül nem ezt választottam.

A program beszerezhető a http://www.colinux.org/ címről kiindulva. A telepítés egy hagyományos windows telepítővel végezhető. A telepítő - munkánkat megkönnyítendő - felajánlja előre elkészített operációs rendszer képfájlok letöltését a telepítés részeként. Az általam használt coLinux verzió 0.7.4, a próbált kernel verziója pedig 2.6.22.18.

3.1.1. Hálózat coLinux alatt

Ahhoz, hogy virtuális gépeink hálózati kapcsolattal is rendelkezzenek telepítenünk kell egy kiegészítő programot is a coLinux mellet. Ez a WinPcap nevű szoftvercsomag. Mire való a winPcap? Ennek magyarázatául a winPcap főoldalán találhatjuk a választ:

"A WinPcap az iparág szabványos eszköze a kapcsolati rétegbeli hálózati hozzáféréshez Windows környezetben: ez teszi lehetővé az alkalmazásoknak, hogy elfogják és továbbítsák a hálózati csomagokat megkerülve a protokoll vermet (protocol stack), illetve további hasznos funkciókat nyújt, beleértve a kernel szintű csomagszűrést, a hálózati statisztikai motorok támogatását és a távoli csomaggyűjtést. "[5]

Az általam próbált verzió a 4.0.2. A telepítő letölthető : http://www.winpcap.org/install/.

[5] A következő internetes cím alapján, saját fordítás: http://www.winpcap.org/default.htm, letöltve: 2009.10.19.

3.2. A vmware player telepítése

Mivel a vmware képes akár 64, akár 32 bites gépek emulálására, és a befogadó operációs rendszerrel szemben is igen rugalmasak így ezen megoldás mellett döntöttem a dolgozatban említett programok kipróbálásának céljából. A vmware player ingyenesen letölthető a következő címről: http://www.vmware.com/products/player/. Az általam használt verzió a 2.5.3 build-185404. A program telepítése windows alatt a megszokott módon zajlik. További érvként hozható fel a vmware mellett, hogy nem csupán egy adott kernellel futtatható virtuális gép, hanem tetszőleges (akár módosított) kernellel is használható, tekintve, hogy teljes számítógépet emulál.

3.3. Teszt operációs rendszer

A teszteléskor használt operációs rendszernek a Debian 5.0 (lenny) rendszert választottam. A szükséges vmware képfájlt a következő címen található előre telepített képfájlok közül választottam ki: http://www.thoughtpolice.co.uk/vmware/.

A tesztrendszerekhez azért választottam 32 bites kernelt, mert úgy vélem, a tesztelendő programok, és azok moduljai egyelőre jobb minőséget és nagyobb lefedettséget biztosítanak 32 bites operációs rendszer alatt, valamint a 32 bites operációs rendszerek elterjedtsége miatt így könnyebb lesz bárki számára a tesztek telepítése. A teszt rendszerekben a root jelszó ezsolt.

3.3.1. Hálózati beállítások

A tesztkörnyezetben egy magánhálózati IP címtartományt használtam. A választás a 10.1.3.0/24 C osztályú hálózatra esett. Mivel a vmwware player konfigurálható úgy is, hogy bridge (híd) módban működjön, így a három virtuális szerver között a az OSI modell 2. rétegében, az adatkapcsolati rétegben történik az összekapcsolás.

Mivel a három virtuális gép egy közös C osztályú hálózatban található, így a kommunikáció a 3. rétegben, tehát a Hálózati rétegben is megvalósul. Bár a legtöbb általam vizsgált rendszernek elég lenne a hálózati rétegben való kapcsolat, bizonyos programok esetében (pl. heartbeat) szükség van arra, hogy az adatkapcsolati rétegben is egy ütközési tartományban helyezkedjenek el a gépek.

3.4. Teljesítménymérés, benchmarking

Ahhoz, hogy összehasonlítható eredményeket kaphassunk a különböző elosztott tárhely megvalósítások között szükséges valamiféle mércét felállítani, mely azonos feltételeket biztosít, és kellően nagy mélységű mérést végez. A mérést abban az esetben, ha egy speciális alkalmazás számára szeretnénk finomhangolni vagy kiválasztani a megfelelő fájlrendszert, speciális kritériumoknak kellene alávetnünk. Mivel jelenleg átfogó képet szeretnék alkotni az egyes fájlrendszerek teljesítményéről, így igyekeztem ennek megfelelő mérési rendszert felállítani. A fájlrendszer tesztelő eszközöket áttekintve három főbb lehetőséget mutatnék be. A méréseket nagyban nehezíti, hogy virtuális gépeket használok, és így a mérések igen pontatlanok a futtató operációs rendszer fájl gyorsítótárazásából adódóan. Ennek ellenére bízom benne, hogy sikerül összehasonlítható adatokat előállítanom.

3.4.1. Egyszerű írási sebesség mérés

Az első dolog, mely kézenfekvő, és minden linux rendszer alatt adott, egy nagyméretű fájl lértehozása, és a létrehozással eltöltött idő mérése. Ezt a módszert nevezhetjük kőkorszakinak, de ennek ellenére mindig kéznél van, nem igényel nagyobb előkészületeket a használata, és tesztelhető vele az is, hogyan reagál a fájlrendszer a nagyobb mennyiségű adattal való megterhelésre. Ha a fájlrendszer hibás, előfordulhat például, hogy a teszt futása alatt nem válaszol a fájllistázási kérésekre. Ez pedig nagy hiba, és ezen egyszerű paranccsal könnyen felderíthető az effajta hibás erőforrás kezelés.

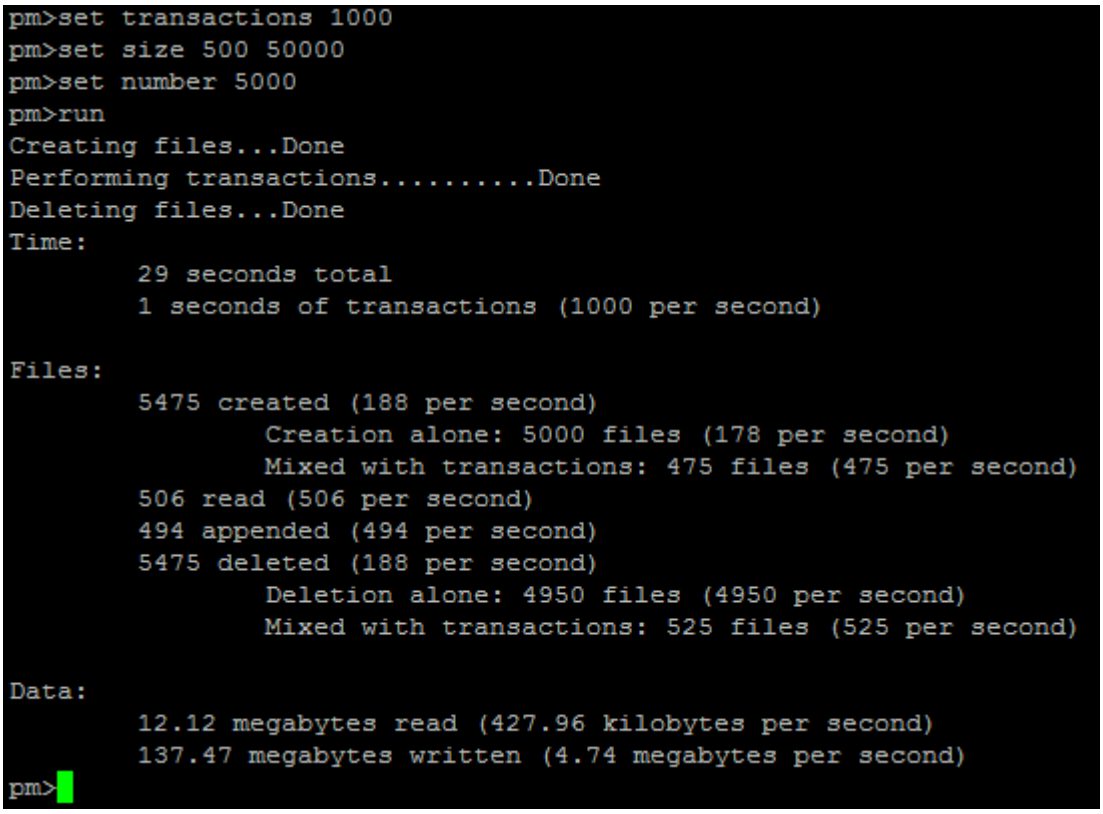
```
# time dd count=5000000 if=/dev/sda2 of=/root/test && rm /root/test
```

3.4.2. Postmark

Sokkal kifinomultabb méréseket végezhetünk a postmark nevű programcsomag segítségével. A program elindításakor egy beépített konzolt kapunk, ahol elvégezhetjük a méréshez tartozó konfigurációt. Miután megadtuk a megfelelő konfigurációs beállításokat, a run parancs segítségével indíthatjuk a teszteket.

1. kép

Teljesítménymérés postmark segítségével



Forrás: saját szerkesztés (2009)

```
set transactions 1000
set size 500 500000
set number 5000
```

A fenti beállítások jelentése a következő: 1000 tranzakció kerül végrehajtásra. A létrehozott teszt fájlok mérete 500 és 500 000 bájt között lesznek véletlenszerűen megállapítva. Illetve 5 000 fájl jön létre egyszerre.

Az eredmények szövegesen jelennek meg. Az eszköz jól használható, és látványos méréseket tudunk elérni segítségével. Tapasztalatom szerint a postmark kitűnő mérőeszköz a különböző elosztott fájlrendszerek mérésére. A tárgyalt fájlrendszerek teljesítményét ezzel az eszközzel végeztem.

3.4.3. IOZone

Az iozone fájlrendszer benchmark eszköz segítségével még részletesebb, és átfogóbb képet alkothatunk az adott fájlrendszer teljesítményét illetően. Szerencsére a debian etch disztribúció tartalmazza az iozone csomagot (non-free szekcióban), így telepítése a szokásos módon történhet

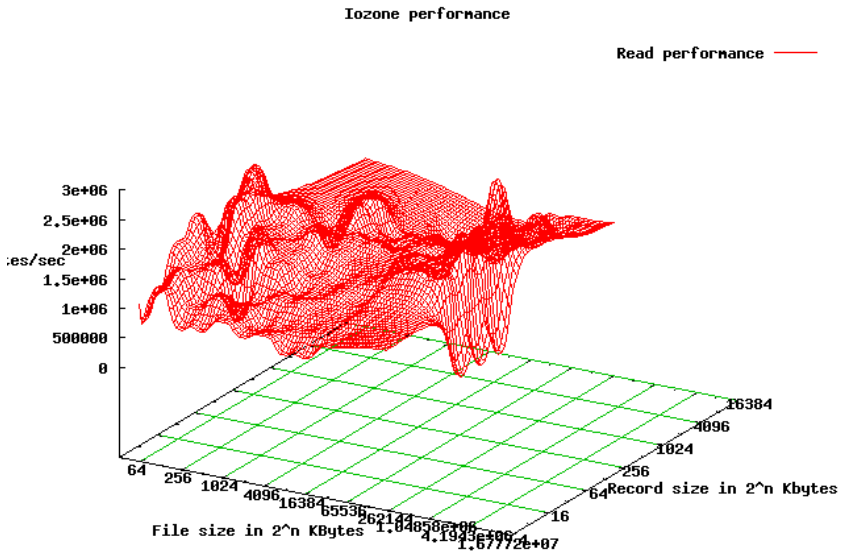
```
#aptitude install iozone)
```

Azért nem választottam a mérésekhez végül ezen eszközt, mert a generált grafikonok bár látványosak, mégsem adnak eléggé átlátható eredményt. Véleményem szerint főképp a fájlrendszerek finomhangolásában nyújthat nagy segítséget. A tesztmérések során az alábbihoz hasonló parancsot használtam:

```
#iozone -a -g 16M | tee -a /tmp/iozone_results.txt
```

2.kép

Teljesítménymérés IOZone segítségével



Forrás: saját szerkesztés (2009)

4. Gyakorlati megvalósítások

A dolgozat gyakorlati részében áttekintésre kerülnek a főbb elosztott tárhely megvalósítási lehetőségek, majd pedig megvizsgálom az elosztott adatbázist lehetővé tevő nyílt forrású szoftvereket. Az elosztott fájlrendszerek alapvető építőkövei bármely elosztott, illetve magas rendelkezésre állású rendszernek.

4.1. Hálózati fájlrendszerek

Elsőként tekintsük át, milyen lehetőségeink vannak egy adott fájlrendszer távoli elérésére. Ez a fajta működés az elosztott fájlrendszerek legegyszerűbb megvalósításai. Működésük arra

épül, hogy egy adott számítógépen található fájlrendszer elérését biztosítja más, kliens számítógépek részére.

4.1.1. NFS

A linux világában a legelterjedtebb távoli fájl elérési módszer az NFS. (Network File System - Hálózati Fájlrendszer) Elterjedtségének köszönhető stabilitása, és széleskörű támogatottsága.

Telepítés

Telepítése igen egyszerű, minden linux rendszerben, sőt legtöbbször alapértelmezetten telepítésre kerül a kliens oldali szoftvercsomag. A szerver telepítése sem igényel nagy erőfeszítéseket. Debian alatt a telepítéshez a következő csomagok szükségesek: nfs-common, nfs-kernel-server. A konfigurálás sem vesz sok időt igénybe. A szerver oldali konfigurációhoz egyetlen fájlra van szükségünk, mely általában a /etc/exports helyen található.

3. kép

NFS export beállítások

```
Fájl: exports           Sor 1 Oszl 0        445 bájt
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes       hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
#
/home/nfsexport         10.1.0.0/24 (rw,sync,no_subtree_check)
```

Forrás: saját szerkesztés (2009)

A fájlban felsorolhatjuk, hogy mely könyvtárakat szeretnénk megosztani, illetve ugyanitt adhatjuk meg, hogy mely IP címekről kapcsolódhatnak szerverünkhöz kliensek, és az egyes IP címekkel azonosított kliensek milyen jogosultságokkal rendelkeznek a megosztások felett, illetve egyéb opciók is megadhatóak. Az IP címek megadásánál használhatunk konkrét címeket, vagy megadhatunk címtartományokat is, illetve lehetőség van helyettesítő karakterek (wildcard) használatára is. Az NFS nek három fontosabb verziója van (NFS2, NFS3, NFS4).

Tekintsük át, milyen előnyökkel és hátrányokkal rendelkezik az NFS3, illetve NFS4.

1. táblázat

Az NFS3 és NFS4 előnyei és hátrányai

NFS3	**NFS4**
Erősségek 1. Viszonylag könnyen alkalmazható 2. Jól illeszkedik a Unix VFS (virtuális fájlrendszer) filozófiába (kivéve a gyorsítótárazást) 3. Könnyen érthető protokoll használata 4. ONC-RPC hitelesítést használ, mely ingyenes és biztonságos 5. Többféle nem Linuxos megvalósítás is létezik (Windows, OS/400, ...) 6. Open Group standard http://www.opengroup.org/bookstore/catalog/c702.htm 7. Elérhető tesztkészlet hozzá http://www.opengroup.org/testing/testsuites/vsx4nfsov.htm 8. speciális szerver teljesítmény mérések http://www.spec.org/benchmarks.html#nfs	Erősségek 1. IETF standardnak megfelelő specifikáció 2. Javított helyreállítás (zár migráció) 3. Jobban támogatja a Windows fájl megosztási szemantikát, mint az NFS v3 4. Biztonságos fájl gyorstárazás 5. Erős autentikációt és integritást biztosít a Kerberos és SPKM-3 használata révén 6. Gazdag hozzáférés szabályozást biztosít a Windows 2000 hez hasonló hozzáférési listák (ACL) révén
Gyengeségek 1. A központi protokoll állapot-mentessége gyorstárazási problémákat okoz 2. Kevés Windowsos NFS kliens létezik 3. A biztonság a Unix felhasználói	Gyengeségek 1. Még mindig viszonylag új - a Linuxos és egyéb megvalósítások már érlelődnek, de még nem széleskörűen elterjedt 2. Nem érzékelhető érdeklődés a

azonosítókon alapul. Ha ezek kijátszhatóak, nincs biztonság 4. Rosszul illeszkedik a Windows rendszer API hoz 5. Nem IETF standard (Sun és NetApp által megfogalmazott információ leírási standard -informális RFC szabvány) 6. Viszonylag gyenge nyílt forrású megvalósítás (a Samba és AFS hez képest) méretezhetőségi problémákkal 7. Ahhoz, hogy a CIFS nek megfelelő legyen sok protokollt kell megvalósítani pl. zár menedzser, mount és port mapping, SunRPC, NIS, ONC kiterjesztések 8. WebNFS bővítések részben implementáltak, mely zavaró 9. Nem támogatja a Unicode, UTF/8, UCS-4, stb. karakterkészleteket	Microsoft részéről 3. Talán túl késő? 4. Komplex

Forrás: http://wiki.linux-nfs.org/wiki/index.php/Comparison_of_NFS_vs._others#NFSv3 alapján, saját fordítás.

Az általam felállított tesztkörnyezetben a debian az NFS4 es verzióját is támogatta. A tesztkörnyezetben a 1. és 2. virtuális szerverek között van beállítva NFS megosztás. A szerver a debian1 gép, míg a kliens a debian2 gép. A debian2 gépen az megosztás becsatolásához a következő parancsot kell kiadjuk:

```
mount 10.1.3.201:/home/nfsexport /mnt -t nfs
```

Authentikáció

A csomópontok IP cím alapján kerülnek azonosításra, mely alacsony szintűnek mondható mind biztonság, mind rugalmasság szempontjából. Lehetőség van viszont a kerberos rendszer felhasználásával ezen authentikáción javítani. Ehhez viszont speciális kiegészítő csomagok szükségesek.

Zárkezelés

Az NFS protokoll (kettes és hármas verziója) nem támogatja a fájl zárolást, de az NFS környezet biztosít egy hasonló mechanizmust, mely az NLM nevet viseli. (NLM - Network Locking Manager - Hálózati Zárkezelő). Amikor egy kliens szerveren NFS fájlrendszer zárolási kérést kap egy fájlra, az NFS távoli eljáráshívás helyett egy NLM távoli eljáráshívás történik. A probléma az, hogy az NFS protokoll állapotmentes, míg a zárkezelés állapottartó folyamat, ezért sok gond származhat a zárak fennmaradásából ha a szerver vagy a kliens, esetleg a hálózat meghibásodik, vagy elérhetetlenné válik.[6]

Teljesítmény

```
Time:
        41 seconds total
        22 seconds of transactions (45 per second)
Files:
        1008 created (24 per second)
                Creation alone: 500 files (26 per second)
                Mixed with transactions: 508 files (23 per
second)
```

[6] A következő internetes cím alapján, saját fordítás:
http://docstore.mik.ua/orelly/networking_2ndEd/nfs/ch11_02.htm, letöltve: 2009.10.19.

```
        504 read (22 per second)
        496 appended (22 per second)
        1008 deleted (24 per second)
                Deletion alone: 516 files (516 per second)
                Mixed with transactions: 492 files (22 per
second)
Data:
        134.44 megabytes read (3.28 megabytes per second)
        290.90 megabytes written (7.10 megabytes per second)
```

Hibatűrés

Az alap NFS3 és NFS4 verziók nem támogatják. Crichlow szerint az NFS4 es verziója által támogatott automatikus befűzés (automount) segítségével lehetőségünk van több fájlszervert is megadni, melyek közül a kliens dönt, hogy mely szerverhez csatlakozik egy adott fájl lekérése esetén, de ez még mindig nem ad megoldást a fájlok replikálására nem is szólva a zárkezelés migrációjáról.

4.1.2. Samba

A Samba a windows "Fájl és nyomtatómegosztás", illetve a "Microsoft Networks Kliens" szolgáltatásokat, valamint sok hasznos segédprogramot tartalmazó programcsomag. A samba tehát kulcsszerepet játszik a windowsos és linuxos világ tárhely szintű összekapcsolásában. Mi jelen dokumentumba a samba nyomtatómegosztási funkcionalitásával nem foglalkozunk, csupán mint fájl megosztási lehetőséget mutatjuk be.[7]

[7] A következő internetes cím alapján, saját fordítás: http://hu.wikipedia.org/wiki/Samba, letöltve: 2009.10.19.

Telepítés

A samba szerver telepítést az 1. virtuális szerveren végeztem el. A kliens telepítése pedig megtörtént a 2. szerveren is, hogy a szükséges mérések elvégezhetőek legyenek. A telepítéskor az alábbi segédletet vettem igénybe: http://www.ctunion.com/node/53

```
[smbmappa]
    comment = Teszt mappa
    browseable = yes
    path = /home/smbshare
    writable = yes
    public = yes
    guest ok = no
    create mask = 0700
    directory mask = 0700
```

```
#aptitude install samba, smbclient, smbfs
```

A samba megosztásait a konfigurációs fájlban adhatjuk meg. Itt jóval több lehetőségünk van a konfigurálásra és finomhangolásra, mint az NFS esetében. Néhány konfigurációs paraméter a Windowsos világ különbözőségeiből adódik, például a fájl és könyvtár létrehozási maszk, de mindenesetre rugalmasabban konfigurálható az NFS nél. A kliens oldali telepítés sem okoz sok fejtörést, debian alatt az smbfs csomag tartalmazza a szükséges kliens komponenseket. A samba teljesítményének méréséhez linux alapú kliensprogramot vettem igénybe. A klinesprogram lehetővé teszi samba megosztások csatolását (mount) fájlrendszerünkbe, így ezt követően az NFS hez hasonló működést érhetünk el. A becsatolás az NFS hez hasonló módon történik:

```
#smbmount //10.2.0.102/smbshare /mnt -o user=smbshare pass=12345
```

Authentikáció

A samba rugalmas authentikációs rendszerrel rendelkezik, ez a PAM (Pluggable Authentication Module - Cserélhető Hitelesítési Modul) használatának köszönhető. Alap esetben a gazda rendszer authentikációs moduljával szinkronban egy saját fájl alapú authentikációs adatbázist használ a beléptetésre. Lehetőség van ugyanakkor ezen authentikációt tetszőleges módon bővíteni (pl. Ldap, adatbázis alapú authentikáció, stb.).

Zárkezelés[8]

Két típusa van a zárkezelésnek, melyeket egy SAMBA szervernek ki kell szolgálnia. Az egyik a rekord zárolás, mely lehetővé teszi egy kliens számára, hogy egy bájt tartományt zároljon egy nyitott fájlon. A másik a kizáró módok, melyek egy nyitott fájlon értelmezhetőek.

A rekord szintű zárolás szemantikája UNIX alatt nagyban különbözik a Windows rekord szintű zárolásától. A Samba 2.2 előtti verziói a közvetlen fcntl() UNIX rendszerhívást próbálták meg használni, hogy megfelelő rekord zárolást valósítsanak meg a különböző Samba kliensek között. Sajnos ez több ok miatt sem lehet teljesen jó megoldás. Az egyik legkézenfekvőbb ok, hogy a Windows ügyfelek akár 2^{32} vagy 2^{64} méretű bájt tartományt is zárolhatnak a kliens operációs rendszertől függően. A UNIX zárolás csupán 2^{31} méretű tartományokat támogat. Így hát nem kivitelezhető, hogy 2^{31}-nél nagyobb kéréseket teljesítsünk. Az említetten kívül számos további különbség létezik, sajnos túl sok, hogy ezeket részleteiben tárgyalhassuk.

A Samba 2.2 és magasabb verziók az alsó szintű operációs rendszertől teljesen függetlenül kivitelezik a rekord zárolást. Amennyiben egy klienstől érkező bájt tartomány zárolási kérés beleesik a 2^{31} tartományba, úgy a Samba továbbítja azt a UNIX rendszer felé. Ezen kívül, a többi zár nem látható a UNIX rétegben számára, azokat nem az operációs rendszer zárkezelésével oldja meg a samba.

Teljesítmény

```
Time:
        123 seconds total
        37 seconds of transactions (27 per second)
Files:
        1008 created (8 per second)
                Creation alone: 500 files (5 per second)
```

[8] A következő internetes cím alapján, saját fordítás: http://www.samba.org/samba/docs/man/Samba-HOWTO-Collection/locking.html, letöltve: 2009.10.19.

```
                Mixed with transactions: 508 files (13 per
second)
        504 read (13 per second)
        496 appended (13 per second)
        1008 deleted (8 per second)
                Deletion alone: 516 files (258 per second)
                Mixed with transactions: 492 files (13 per
second)
Data:
        134.44 megabytes read (1.09 megabytes per second)
        290.90 megabytes written (2.37 megabytes per second)
```

Amint látható, minden tekintetben jóval gyengébb eredményt produkált az NFS nél.

Hibatűrés

Jelenleg semmilyen eszközzel nem támogatja a hibatűrő működést.

4.2. Elosztott fájlrendszerek

Az eddigiben vizsgált módszerek nem fájlrendszereket takartak, sokkal inkább távoli fájl elérést, illetve a kliens oldalon ennek transzparenssé tételéhez fájlrendszer emulációt. A következőkben vizsgáltak ezzel szemben már főként elosztott fájlrendszerek.

4.2.1. OpenAFS[9]

Az AFS egyszerűvé teszi az emberek számára a fájlokon való együttműködést, nem számít azok hol vannak helyileg. Az AFS felhasználóknak ugyanis nem kell tudniuk, mely gép tárolja a fájlt, az adminisztrátorok pedig átmozgathatják a fájlokat gépről gépre a felhasználók hozzáférésének megszakítása nélkül.

A felhasználók mindig ugyanazon elérési út segítségével azonosíthatnak egy fájlt, és az AFS automatikusan megtalálja a megfelelő fájlt, úgy, mint ahogy az egy helyi fájlrendszeren történik. Az AFS megkönnyíti a fájlmegosztást, de ez nem megy az osztott fájlok biztonságának rovására. Ezt kifinomult védelmi sémák biztosítják.

Telepítés[10]

Mivel az OpenAFS megkövetlei a Kerberos authentikációt, első lépésként ennek beállítását mutatom be. (Master password: ezsolt)

```
#aptitude install krb5-{admin-server,kdc}
```

```
#aptitude install openafs-client, openafs-dbserver, openafs-
fileserver, openafs-mopdules-source openafs-krb5
#aptitude install module-assistant
#m-a prepare openafs
#m-a a-i openafs
```

A telepítéshez egy, az OpenAFS klienshez szükséges kernel modult is telepítenünk kell. Szerencsére a module-assistant nevű debian csomagban található eszközök leveszik a vállunkról a modulépítés terhét, és automatizáltan elvégzi a szükséges műveleteket, hogy a forráskódból előálljon az aktuális kernelhez szükséges debian csomag, majd ezen csomagot fel is telepíti, így a kernel modul elérhetővé válik. Ezt láthatjuk a 2. 3. és 4. lépésben.

[9] A következő internetes cím alapján, saját fordítás: http://docs.openafs.org/UserGuide/ch01.html#HDRWQ3, letöltve: 2009.10.19.
[10] A következő cikk alapján: http://www.debian-administration.org/article/OpenAFS_installation_on_Debian, letöltve: 2009.10.25.

Nehézséget jelent viszont, hogy kernelfrissítés esetén ezen műveleteket újra el kell végezzük, és újra fel kell építenünk a megfelelő kernel modulokat. Ez az OpenAFS negatívumaként említhető, mert megnehezíti a szofver karbantartását.

Zárkezelés[11]

Linux környezetben az OpenAFS illeszkedik a szerver operációs rendszer zárkezelési mechanizmusához.
Viszont sok windowsos alkalmazás (pl. Microsoft Office) igényli a bájt tartomány szintű fájlzárolást akár a szimultán fájlhozzáférések megakadályozására, akár szignál mechanizmusként. Az OpenAFS Windowsos változata az 1.5 (vagy magasabb verzióknál) megvalósítja a bájt tartomány zárolást a CIFS-AFS átjáró segítségével. A bájt tartomány zárolás megvalósítására az AFS opcionális (advisory) fájlzárolását használja, hogy szimuláljuk a Windows kizáró (mandatory) zárkezelését. Amikor egy alkalmazás megnyit egy fájlt, egy zárt helyez rá az AFS, hogy ezzel jelezze, hogy a fájl használatban van. Ha a zár egy írási zár, más alkalmazások nem férhetnek hozzá a fájlhoz ugyanazon gépről. A más gépeken futó programok a teljes AFS zárat látják, és nem férhetnek hozzá a fájlhoz.

Gyorstárazás

Igen kifinomult gyorstárazással rendelkezik. Egy külön könyvtár adható meg a konfigurációban a gyorstárazáshoz, mely természetesen csatolható akár egy külön erre a célra használt, gyorsabb működésű háttértárhoz is, vagy akár a ramfs nevű memória alapú fájlrendszer is használható erre a célra.

[11] A következő internetes cím alapján, saját fordítás:
http://docs.openafs.org/ReleaseNotesWindows/ch03s19.html, letöltve: 2009.10.25.

Authentikáció

PAM on keresztül kerberos authentikációt javasol alap telepítés esetében a szolgáltatás jellegű authentikációhoz. A kliensek csatlakozásánál, illetve a fájlok elérhetőségének konfigurálása igen rugalmasan, jól paraméterezhető ACL (Acces Controll List - Hozzáférést Szabályozó Lista) révén állítható a felhasználói igényeknek megfelelően.

Hibatűrés

Az AFS csak olvasható fájlreplikációkkal biztosít némi redundanciát azon fájlok hibatűrő eléréséhez, melyek nélkülözhetetlenek, vagy nagy gyakorisággal kerülnek olvasásra. Sajnos ez egy csak olvasható replikáció, tehát nem nyújt transzparens működést. Nem biztosítható vele hibatűrő működés általánosságban.

E mellet a rendelkezésre állás javítását segíti a gyorstárazási is, ugyanis ha valamely fájlszerver nem elérhető, a gyorstárban található legutóbbi verzió olvasásra még mindig használható. Azt azonban nem jelenthetjük ki, hogy ez egy általános célú hibatűrő rendszer megvalósítására elegendő lenne.

4.3. Párhuzamos működésű hibatűrő elosztott fájlrendszerek

„Az elosztott fájlrendszerek, amelyek lehetnek párhuzamosak és hibatűrők, és a szerverek közötti adat-replikációval érik el a nagyobb teljesítményt, és a nagy megbízhatóságot, és biztosítják az adat integritást. Még ha egy szerver ki is esik, akkor sincs adatvesztés, és a működés folyamatossága nem szakad meg. Ezen fájlrendszereket a HCP és a magas rendelkezésre állású klaszterek is használják.”[12]

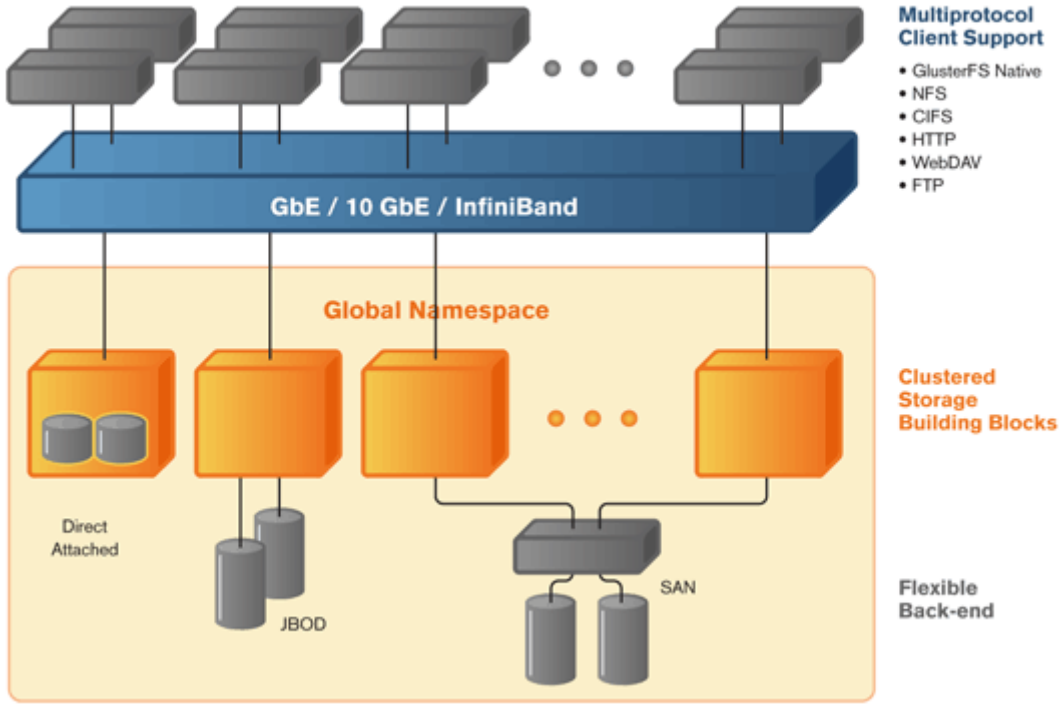
[12] A következő internetes cím alapján:
http://hu.wikipedia.org/wiki/Fájlrendszerek_listáa#ElosztottPárhuzamosFájlrendszerek, letöltve: 2009.10.28.

4.3.1. GlusterFS

A GlusterFS egy szabad / nyílt forráskódú fürtözött fájlrendszer, amely képes több petabájtos adatmennyiség kezelése skálázható módon, több ezer ügyfél kiszolgálása mellett. A GlusterFS klaszterek közös tároló építőkockákat, lemez és a memória-erőforrások kezelését és az adatok egy egységes, globális névtérként való kezelését végzik.[13]

4.kép

A glusterfs szerkezeti felépítése



Forrás: http://www.gluster.com/products/index.php, letöltve: 2009.10.28.

A GlusterFS egyik legfontosabb előnye a moduláris felépítés, mely lehetővé teszi, hogy a modulok a felhasználói igényektől függően egymásra épüljenek. A GlusterFS segítségével gyorsan beállítható egy önálló kiszolgáló rendszert, mely később az igények növekedésével tetszőlegesen bővíthető.

[13] A következő alapján, saját fordítás: http://www.gluster.com/community/documentation/index.php/GlusterFS_Introduction#Overview, letöltve: 2009.10.21.

A GlusterFS filozófiája alapvetően eltér az eddig megismert fájlrendszerektől. Az első legnagyobb különbség, hogy a szerver és a kliensek is felhasználói módban futnak, nem pedig a rendszermagba épülnek be. Ezen működést a linux kernelben megtalálható fuse modul teszi lehetővé. (fuse= Filesystem in Userspace - fájlrendszer felhasználói módban, telepjtés aptitude install libfuse2), mely segítségével felhasználói jogosultságokkal is becsatolhatunk egy fájlrendszert. A fuse modulon keresztül történő fájlrendszer becsatolás manapság egyre népszerűbbé válik. Elérhető egyre több fájlrendszer fuse verzióban is, többek közt az NFS ben is lehetőség van felhasználói módban futtatni a szervert. (A csomag neve nfs-user-server.)

A fuse mód azért számít majdhogynem forradalminak, mert jócskán megkönnyíti a különböző rendszermagokhoz történő csatlakozást, illetve a szoftver frissítését, ugyanis nem a rendszermagba épül be a program. Ez természetesen azzal a hátránnyal jár, hogy a teljestímény valamennyivel gyengébb lesz. Ezt kiküszöbölendő a GlusterFS készítői kiadták saját fuse moduljukat, mely nagyobb teljesítményt és áteresztő képességet biztosít. Ezt akkor érdemes telepíteni, ha már kipróbáltuk a fájlrendszer nyújtotta lehetőségeket, és szeretnénk éles környezetbe ültetni, és használni.

Másik fontos különbség az eddig megismert fájlrendszerekhez képest, hogy a glusterfs háttértárolóként (backend) nem közvetlenül a blokkegységet használja, hanem egy már meglévő fájlrendszert használ erre a célra. Ráadásul opcionálisan választható a Berkeley DB adatbázis kezelő is, mely egy nagy teljesítményű beágyazható adatbázis kezelő. Ennek segítségével a kis fájlok is igen hatékonyan tárolhatóak, és nem okoz problémát a nagyméretű könyvtárak kezelése sem, mely sok egyéb linux alapú fájlrendszerek gyenge pontja.

Mivel a glusterfs nem található meg a debian csomaglistában, mert viszonylag új szoftvernek számít, így kénytelenek vagyunk forráskódból magunk fordítani azt (illetve régebbi verziók megtalálhatóak a testing és unstable debianban, de jobban járunk ha lefordítjuk). A fordítás után, hogy minden szerverre fel tudjuk telepíteni, és ne kelljen mindenhol újra a fordítással megbirkóznunk, a checkinstall nevű programot hívjuk segítségül, mely a make install parancsot helyettesíti, és a telepítés helyett egy telepíthető debian csomagot készít számunkra, mely már bármely hasonló debian rendszerre könnyen telepíthető. (dpkg -i paranccsal) A fordításokat és a telepítőt a /usr/src könyvtárban találhatjuk meg az első virtuális szerveren.

A GlusterFS-sel létrehozott fürtünk elérésére több lehetőségünk is van. Egyrészt becsatolhatjuk azt a glusterfs saját csatolóján keresztül. Másrészt viszont a glusterfs biztosítja számunkra azt is, hogy NFS, SAMBA vagy akár WebDAV protokollon keresztül érjük el, úgy, hogy a becsatolt fájlrendszert NFS vagy SAMBA szerverrel továbbdelegáljuk, vagy a glusterfs WebDAV csatolóját használjuk.

Az általam felállított topológiában mindhárom gép egyszerre kliens és szerver is valamint, azért hogy be tudjam mutatni a hibatűrő működést mindhárom gépen tárolásra kerül minden adat. Több gép esetén szerencsésebb talán, ha a redundanciát kisebbre állítjuk, hogy ezzel helyet spóroljunk, és növeljük a teljesítményt is de ebből is kitűnik a glusterfs remek, moduláris felépítésének előnye, hogy lehetőségünk van tetszés szerinti redundancia meghatározására. A redundáns működést az AFR (Automatic File Replications - automatikus fájl replikáció) fordító (translator) biztosítja. A fordítókról egy kicsit bővebben is szót ejtenék, mert ez által kaphatunk bepillantást a glusterfs működésének zsenialitásába.

Mivel a becsatoláshoz a glustefs saját kliensét használom, így célszerűnek láttam az AFR-t a kliens oldalon elhelyezni. Ez azért előnyösebb a szerver oldali megoldásnál, mert ha szerver oldalon lenne, a kliens nem tudná, hogy mely szerverekhez csatlakozhat hiba esetén.

Fordítók (Translators)

A glusterfs által nyújtott fordítókat a következő kategóriákba sorolhatjuk:

- protocol - Ide tartozik a server illetve client translator. Ezek biztosítják a IP vagy Infiniban Alapó kapcsolódást
- cluster - Ebbe a csoportba azon fordítók tartoznak, melyek az elosztott működéshez szolgáltatnak eszközrendszert. Ide tartozik a distribute (vagy korábbi nevén DHT), mely a megadott csomópontok között elosztott fájltárolást tesz lehetővé, a könyvtárszerkezetet pedig az egyes filenevekhez rendelt hash kódokkal osztja el. Hasonló funkcionalitást biztosít a nufa fordító is, de ennél egy adott szerverről érkező kliensek az adott szerveren kerülnek tárolásra. Ezen csoportba tartozik még a stripe fordító is, mely főként nagy fájlok több szerveren történő elosztásában játszik szerepet, illetve a replicate, mely pedig a fájlok többszörözésében, redundáns tárolásában nélkülözhetetlen.

- performance - Az ebbe acsoportba tartozó fordítók mind valamilyen jellegű teljesítmény javítást, illetve finomhangolást tesznek lehetővé. A readahead illetve writebehind fordítókat általában alacsony szinten, a háttér fájlrendszerhez közel használják, hogy segítéségvel optimalizálni lehessen az írási és olvasási műveleteket, vagy a hálózati szinten használját a hálózati forgalom optimalizálása végett. Az io-cache és io-threads fordítók pedig a többszálúság és az IO gyorstárazás finomhangolására használhatóak.
- egyéb - A többi fordító kisebb területeket fednek le, úgy mint nyomkövetés, zárkezelés, szűrés illetve titkosítás.

Zárkezelés

A zárkezelést a posix illetve a locks nevű fordító segítségével biztosítja. Ezen fordító pedig a mögöttes tároló zárkezelési mechanizmusát használja.

Teljesítmény

A teljesítmény a kívánt architektúra függvénye. Természetesen enm várható el, hogy AFR rel jobb írási sebességet érjünk el, mint hagyományos elérés esetében, mivel az írási művelet befejeztét meg kell várjuk az AFR blokk minden tagján. Ha viszont célunk a nagyobb írási sebesség, akkor nagy fájlok esetén a stripe, sok kis fájl esetében pedig a DHT fordítót használhatjuk, melyek igen jó teljesítményt biztosítanak.

```
Time:
        1810 seconds total
        1424 seconds of transactions (0 per second)
Files:
        1008 created (0 per second)
                Creation alone: 500 files (1 per second)
                Mixed with transactions: 508 files (0 per
second)
        504 read (0 per second)
        496 appended (0 per second)
        1008 deleted (0 per second)
                Deletion alone: 516 files (172 per second)
                Mixed with transactions: 492 files (0 per
second)
Data:
        134.44 megabytes read (76.06 kilobytes per second)
        290.90 megabytes written (164.57 kilobytes per second)
```

Gyorstárazás

Mind szerver, mind kliens oldalon többféle lehetőségünk van a gyorstárazásra, mely a performance (teljesítmény) fordítók használatával érhető el. A gyorstárazáson kívül lehetőségünk van még további finomhangolásokra is, például kis fájlok blokkosított átvitelére, vagy előre olvasásra (readahead), illetve utó-írásra (writebehind). Ezen eszközök segítségével alkalmazásainknak megfelelően finomhangolhatjuk a fájlrendszert.

Authentikáció

A szerverek a klienseket IP címük alapján azonosítják. A szervereknél megadható, hogy mely kötetek (volume) mely IP címekről érhetőek el. A kliensek nem biztosítanak extra védelmi vonalat a fájlok elérésének korlátozására.

Hibatűrés

A glusterfs konfigurálható szintű hibatűrést biztosít az AFR (Automatic File Replication - Automatikus Fájl Többszörözés) fordító segítségével. Az AFR gondoskodik róla, hogy a beállított alsóbbrendű csomópontok mindegyikén tárolásra kerüljenek a fájlok. A fájlok épségének biztosításához lusta öngyógyítást (lazy self healing) alkalmaz, mely azt jelenti, hogy egy adott fájl elérése esetén ellenőrzi a fordító, hogy az adott fájl minden csomóponton megtalálható-e, és a legfrissebb verzió érhető-e el. Ha nem, akkor pedig gondoskodik a legfrissebb verzió szétosztásáról.

4.4. Osztott tárolóeszköz alapú filerendszerek

„Az osztott diszkes fájlrendszerek (gyakran nevezik *osztott tárolójú fájl rendszer*nek, vagy még *cluster vagy fürt fájl rendszer* néven is ismert) elsősorban a tároló hálózatokban (NAS - Network Attached Storage - Hálózat Csatolású Tárkezelés) használatos, ahol minden csomópont (node) közvetlenül hozzáfér a tárolókhoz. Ez a megoldás biztosítja, hogy egy csomópont kiesése nem befolyásolja egy másik csomópont adathozzáférését. Az osztott diszkes fájlrendszereket általában ú.n. "magas rendelkezésre állású klaszterek" használnak egy RAID megoldással kiegészítve. Az osztott diszkes fájlrendszereket legfeljebb 64 vagy 128 csomópont esetén használják. Az osztott diszkes fájlrendszerek lehetnek szimmetrikusak,

ekkor meta-adatok kerülnek szétosztásra a csomópontok között, illetve lehetnek aszimmetrikusak, amikor a meta-adatokat központi meta-adat szerver(ek) tárolják(k).”[14]

4.4.1. GFS (Global FileSystem)[15]

A GFS és GFS2 abban különböznek az elosztott (distributed) fájlrendszerektől (mint például AFS, Coda, InterMezzo), hogy a GFS és GFS2 minden csomópont (node) számára közvetlen, konkurens hozzáférést tesz lehetővé ugyanazon osztott blokktároló eszközhöz.Ráadásul a GFS és GFS2 helyi fájlrendszerként is alkalmazhatóak.

A GFS nem rendelkezik csatlakozás nélküli üzemmóddal, és nincsenek kliens és szerver szerepkörök. Egy GFS klaszterben minden csomópont egyenrangú tagként (peer) viselkedik. A GFS alkalmazása egy klaszterben megköveteli olyan hardvert alkalmazását, mely lehetővé teszi az osztott tároló elérését, illetve olyan zárkezelőt, mely kezeli a tárolóeszközhöz történő hozzáféréseket. A zárkezelő egy külön modul, így a GFS és GFS2 használhatják a DLM -et (Distributed Lock Manager - Elosztott Zár Menedzser) klaszter kialakításához, vagy az "nlock" zárkezelő menedzsert helyi fájlrendszerekhez. A GFS régebbi verziói szintén támogatják a GLUM -ot, egy szerver alapú zárkezelőt, mely redundanciát biztosít.

A GFS rendszer részletes bemutatását és telepítését nem végeztem el, mert nem áll rendelkezésemre megfelelő hardver, melyet a GFS osztott tárolóeszközeként tudnék használni. A leírások alapján a GFS képes üzemelni a GNBD (Global Network Block Device - Globális Hálózati Blokk Eszköz) segítségével általános célú blokkeszközökön is, illetve a DRBD nyílt forrású szoftveres hálózati RAID1 megvalósítás aktív-aktív módjában is. Ezen megoldások vizsgálata azonban sajnos meghaladják jelen dolgozat kereteit.

---

[14] A következő internetes cím alapján: http://hu.wikipedia.org/wiki/Fájlrendszerek_listáa#ElosztottPárhuzamosFájlrendszerek, letöltve: 2009.10.29.

[15] A következő internetes forrás alapján, saját fordítás: http://en.wikipedia.org/wiki/Global_File_System, letöltve: 2009.10.16., illetve saját fordítás elhelyezve: http://hu.wikipedia.org/wiki/Global_File_System

4.5. Peer-To-Peer fájlrendszerek

A Wikipedia szerint a **peer-to-peer** vagy **P2P** paradigma lényege, hogy az informatikai hálózat végpontjai közvetlenül egymással kommunikálnak, központi kitüntetett csomópont nélkül. A peer-to-peer fogalom két hasonló, de célját tekintve mégis eltérő fogalomkört is takar: a számítógépek egyenrangú technológiai szintű kapcsolódási módját egy helyi hálózaton, valamint valamilyen célból közvetlenül kapcsolódó szoftver megoldások működési elvét. A közvetlen kapcsolat hibatűrőbb felépítést, skálázhatóságot jelent. Hátrányai: a nehezebb adminisztráció, az erőforrások pazarló használata, a nehezebb megvalósíthatóság.[16]

Úgy vélem, a jövő ebbe az irányba mutat, és a P2P fájlrendszereknek nagy jelentőségük lesz a közeljövőben.

Véleményem szerint könnyen kiküszöbölhetőek hátrányaik, és a felépítésben rejlő robusztus és dinamikusan skálázható működésnek köszönhetően várható a széleskörű elterjedés. Sajnos jelenleg még igen kevés megvalósítás érhető el a szabad szoftverek piacán, és a meglévő programok is véleményem szerint kiforratlanok, és nem rendelkeznek széles körű támogatottsággal. Sajnos a dolgozat készítés időpontjában nem állt még rendelkezésre működőképes, stabil peer-to-peer fájlrendszer megvalósítás.

[16] A következő internetes forrás alapján: http://hu.wikipedia.org/wiki/Peer-to-peer, letöltve: 2009.10.18.

5. Adatbázis elosztási lehetőségek

A különböző relációs adatbázisok, csakúgy, mint a fájlrendszer szintén nélkülözhetetlen építőkövei a legtöbb alkalmazásnak. Az alkalmazás növekedtével gyakran előfordul, hogy a megnövekedett terhelést már nem képes egyetlen szerver kiszolgálni. Felmerül az igény, hogy ahelyett, hogy az adatbázis szervert próbálnánk minél nagyobb teljesítményű hardverekkel javítani, egy olyan megoldást keressünk, mely lehetővé teszi az adatbázisok elosztását. Az adatbázisok elosztására két technikát fogunk megvizsgálni, és ezeket a MySQL nyílt forrású relációs adatbázis kezelő segítségével mutatom be.

5.1. Replikációk

Az egyik lehetőség a replikációk készítése. A wikipedia meghatározása alapján „az **informatikában a replikáció** az információk megosztásának folyamata mely lehetővé teszi a konzisztencia biztosítását redundáns erőforrások között, úgy mint szoftver és hardver komponensek. **Adat replikációról** beszélünk, ha ugyanazon adat kerül tárolásra több tárolóeszközön, vagy **számítási replikáció**, ha ugyanazon számítási feladat kerül futtatásra egyidőben.”[17]

Az adatbázis replikációk lehetővé teszik, hogy egy megkülönböztetett (master - mester) szerveren lefuttatásra kerülő SQL utasítások minden alárendelt (slave - szolga) szerveren futtatásra kerüljön. Ez a technika általában az adatbázis szerver naplózó szolgáltatására alapul (pl. MySQL), de előfordulnak trigger alapú megoldások is (pl. PostgreSQL).

A replikációk hátránya, hogy minden módosító utasítást a mester szerveren kell végrehajtani, mely lehetőség bizonyos esetekben nem megvalósítható. Amennyiben viszont az alkalmazás, melyet ezen módon szeretnénk terheléselosztás vagy magas rendelkezésre állás biztosítása érdekében elosztottá tenni támogatja a lekérdező (DQL) illetve adatmódosító (DML) utasítások különbontását, úgy ezen megoldás viszonylag egyszerűen beállítható és kis erőforrás igényű megoldást biztosíthat.

[17] A következő internetes cím alapján: http://en.wikipedia.org/wiki/Replication_%28computer_science%29, letöltve: 2009.10.28.

5.1.1. MySQL replikáció

A MySQL igen népszerű szolgáltatása a replikációk kezelése. Ezt a binlog (bináris napló) szolgáltatás révén teszi lehetővé. A mester szerver minden SQL utasítást naplóz a saját mester naplójába, majd a kapcsolódó alárendelt adatbázis kezelők ezen utasításokat a hálózaton keresztül áttöltve lefuttatják saját adatbázisaikon. Amennyiben valamilyen oknál fogva egy szolga gép nem képes lefuttatni az áttöltött utasításokat, egy különálló naplófájlba gyűjti fel azokat, és a hiba elhárultával lefuttatja a felgyűlt kéréseket.

A replikációt az első és második számú szerverek között állítottam be. Az első szerver a mester, míg a második a szolga szerepkörben működik. A replikáció felállítását az alábbi leírás segítségével végeztem el: http://dev.mysql.com/doc/refman/5.0/en/replication-howto.html.

Első lépésben létrehoztam egy új adatbázist 'replikacio' néven, és feltöltöttem néhány tesz adattal. A létrehozáshoz használt SQL utasítás:

```
CREATE TABLE `felhasznalo` (
  `id` int(10) unsigned NOT NULL auto_increment,
  `user` varchar(30) default NULL,
  `pass` varchar(30) default NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;
INSERT INTO `felhasznalo` VALUES
(1,'ezsolt','12345'),(2,'viki','12345');
```

A következő lépés, hogy létrehozzuk azon felhasználót, mely segítségével a szolga adatbázis kapcsolódhat a mesterhez. Ha több szolga adatbázist is szeretnénk beállítani, célszerű mindegyikhez egy külön felhasználót létrehozni.

```
mysql> GRANT REPLICATION SLAVE ON *.* TO 'ezsolt2'@'%'
IDENTIFIED BY 'jelszo';
```

A százalék jel (%) a mysqlben egy helyettesítő karakter (wildcard), mely tetszőleges mintára illeszkedik. Az általam létrehozott felhasználó nem túl biztonságos, mivel tetszőleges IP címről engedi a csatlakozást. Ez természetesen nagy biztonsági hiányosságot jelent, de az egyszerűbb kezelhetőség kedvéért, illetve azért hogy a tesztkörnyezet más IP címek esetében is működőképes legyen ezen kevésbé biztonságos megoldást választottam. Ezen lépésnél már az is látható, hogy a mysql replikációk authentikációs rendszerként a mysql saját authentikációs rendszerét használják.

Következő lépés, hogy megadjuk a mysql konfigurációs állományában, hogy használja a binlog szolgáltatást, illetve beállítjuk a szerver egyedi azonosítóját. Az egyedi azonosítóra azért van szükségünk, hogy a replikációs rendszeren belül az egyes szerverek egyértelműen azonosíthatóak legyenek. Ez az azonosító egyedi kell hogy legyen minden résztvevő szerver esetében.

```
[mysqld]
server-id               = 1
log_bin                 = /var/log/mysql/mysql-bin.log
expire_logs_days        = 10
max_binlog_size         = 100M
binlog_do_db            = replikacio
```

A konfigurációs állományban megadható az is, hogy hány napra visszamenőleg őrizze meg a szerver a binlogokat. A binlog a replikációk kezelésén felül igen hasznos akkor is, ha megsérül adatbázisunk, vagy törlésre kerülnek bizonyos adatok. Segítségével egy korábbi biztonsági másolat segítségével tetszőleges időpontra visszaállítható az adatbázis állapota. Megadható továbbá, hogy mely adatbázisokat szeretnénk a binlogban szerepeltetni.

Most, hogy a szerver beállításai megtörténtek, elkezdhetünk foglalkozni a szolgákkal. A szolga szerepkörben működő szervereknek is be kell állítsunk egy egyedi azonosítót. A szolga gépeken nem kötelező a binlog engedélyezése, de a fentebb említett biztonsági rendszerként jó hasznát vehetjük. Ezen felül pedig lehetőségünk vagy egy összetettebb replikációs topológia kialakítására is, melyben egy adott szolga gép lehet más szerverek szemszögéből mester.

Egy új replikáció inicializálásának első lépése a mester szerveren található státusz ellenőrzése. Fel kell ugyanis jegyezzük, hogy jelenleg milyen pozíción áll a mester szerveren a binlog. Ezzel egyidőben, vagy ez előtt a lépés előtt meg kell akadályozzuk, hogy a mester szerveren a mester szerveren található adatok mentése alatt további tranzakciók futhassanak le.

```
mysql> FLUSH TABLES WITH READ LOCK;
mysql> show master status;
+------------------+----------+--------------+---------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_D
+------------------+----------+--------------+---------------+
| mysql-bin.000001 |      594 |
+------------------+----------+--------------+---------------+
1 row in set (0.00 sec)
```

Ezt követően pedig elkezdhetjük a jelenlegi adatok mentését. Erre két lehetőségünk is van. Az egyik, hogy a bináris adatokat átmásoljuk a szerverek között, a másik pedig a mysqldump segédprogram alkalmazása. A mentés elkészültével fel kell oldanunk az írási zárat, hogy ne akadályozzuk tovább az adatbázisszerver működését. A mester adatbázisszerver általában kritikus fontosságú, és sok esetben nagy költségekkel jár a zár elhelyezése, és az ebből adódó szolgáltatási szünet. Ötletes megoldás a leállás minimalizálásához, ha olyan filerendszeren helyezzük el az adatbázisszerver adatait, mely képes pillanatképet (snapshot) készíteni a fájlrendszer egy adott időpontban fennálló állapotáról. Ilyen lehetőséget biztosít az lvm (logical volume manager).

```
mysql> UNLOCK TABLES;
```

Az adatok átmásolását követően már nincs más dolgunk, mint a szolga szervereken inicializáljuk a replikációs adatokat, és elindítjuk a replikációt

```
mysql> CHANGE MASTER TO
    -> MASTER_HOST = '10.1.3.201',
    -> MASTER_USER = 'ezsolt2',
    -> MASTER_PASSWORD = 'jelszo',
    -> MASTER_LOG_FILE='mysql-bin.000001',
    -> MASTER_LOG_POS=594;
mysql> START SLAVE;
```

Nos, ezzel a lépéssel el is indult a replikációnk. Mostantól minden SQL utasítás, mely bekerül a mester szerver bináris naplójába, futtatásra kerül a szolga szerveren is. Ez a megoldás igen hasznos, de mégsem teljesértékű. A hibatűrést ugyanis nem biztosítja felépítéséből adódóan, és a lekérdezések nem elosztott módon futnak le. A mester szerver pedig szűk keresztmetszetet jelent a skálázhatóság és a magas rendelkezésre állás szempontjából is. A következőekben vizsgált módszerren keresztül viszont bemutatásra kerül egy teljesértékű, elosztott adatbázis kezelési megoldás.

5.2. Adatbázis klaszterek

A replikáció mellett a klaszterek kialakítása nyújthat megoldást az adatbázisok elosztott működésének és a magas rendelkezésre állás kérdésének megválaszolásában. Az adatbázis klaszterek a replikációval szemben valóban elosztott működést biztosítanak. Nincs szükség a DML és DQL utasítások elkülönítésére, és a maguk a lekérdezések is elosztott módon kerülnek végrehajtásra. Ez jóval nagyobb teljesítményt biztosít a replikációkkal szemben, illetve kiküszöböli a mester szerver által fennálló hibalehetőséget (single point of failure).

5.2.1. MySQL klaszter

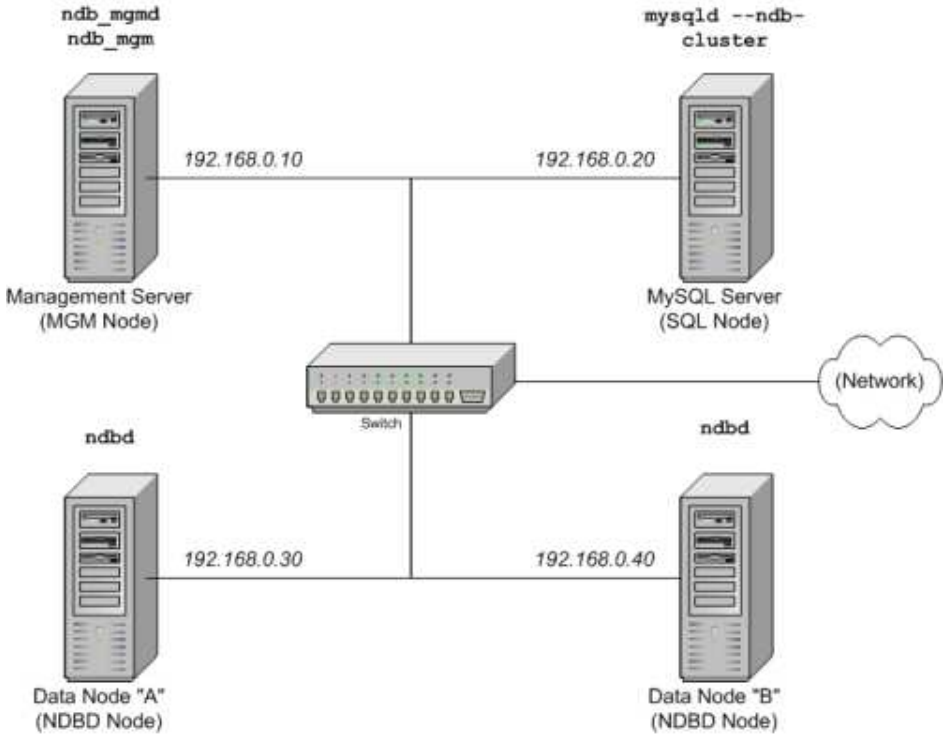
A MySQL Cluster egy olyan technológia, amely osztatlan klaszterezési képességekket biztosít a MySQL adatbázis kezelő rendszernek. Elsőként 2004 novemberébe került be a MySQL 4.1-es verzióba. Arra tervezték, hogy magas rendelkezésre állást és nagy teljesítményt nyújtson, közel lineáris skálázhatóság mellett. A MySQL Cluster egy új tároló motor segítségével került megvalósításra a MySQLben, melyet NDB-nek vagy NDBCLUSTER-nek hívnak (NDB Network Database - Hálózati Adatbázis).[18]

A MySQL Cluster telepítése

A MySQL Cluster telepítése során mindhárom szervert felhasználom. Igaz, hogy két szerverrel is kialakítható a klaszter, a MySQL Cluster tudathasadásos állapotának (split brain) elkerülése végett úgy döntöttem, mindhárom szerverre telepítésre kerül a szoftver. A hármas számú szerver lesz a management szerver, mely biztosítja az adat csomópontok vezérlését.

5.kép

MySQL Cluster felépítése sémája



Forrás: http://dev.mysql.com/doc/refman/5.0/en/images/multi-comp-1.png, letöltve: 2009.10.28.

[18] A következő internetes cím alapján, saját fordítás: http://en.wikipedia.org/wiki/MySQL_Cluster, letöltve: 2009.10.28.

A MySQL Cluster telepítése során találkozunk olyan fogalmakkal, melyek egy egyedülálló MySQL adatbázis szerver esetében nem kerülnek terítékre. Az következő részben elsőként ezen fogalmakkal ismerkedünk meg, majd pedig bemutatásra kerül a Cluster telepítése.

A fogalmak meghatározását az alábbi internetes leírás fordításával készítettem el: http://dev.mysql.com/doc/refman/5.0/en/mysql-cluster-nodes-groups.html.

Menedzsment csomópont: Ezen típusú csomópont szerepe a többi csomópont kezelése egy MySQL Clusterben, olyan funkciók biztosításával, mint konfigurációs adatok biztosítása és a csomópontok indítása és leállítása. Mivel ezen csomópont kezeli más csomópontok beállításait, ennek a csomópontnak kell minden más csomópont előtt, elsőként elindulnia.

SQL csomópont: Ez csomópont, mely az adatok elérését szolgálja. A MySQL Cluster esetében egy SQL csomópont egy hagyományos MySQL servert jelent, mely használja az NDBCLUSTER tároló motort.

Adat csomópont (Data Node): Egy ndbd processzus, mely replikákat (replica) tárol. A replikák a másolatai egy partíciónak, mely az adott csomópont csoportjához tartoznak. Minden adat csomópont külön gépen kell elhelyezkedjen. Bár lehetséges több ndbd processzus futtatása egy gépen, az ilyesfajta beállítás nem támogatott. Általában a csomópont és adat csomópont fogalmakat felváltva alkalmazzák, mialatt ndbd processzusokról beszélünk.

Csomópont csoport (Node Group): Egy csomópont csoport egy vagy több csomópontból áll, és partíciókat, vagy replikációk halmazát tárolja. A csomópont csoportok száma egy MySQL Clusterben nem közvetlenül konfigurálható, ez az adat csomópontokon és a replikációk számától dől el. (NumberOfReplicas paraméter)

[csomopont_csoportok_szama] = adat_csomopontok_szama / ReplikációkSzáma

Így, egy MySQL Cluster négy adat csomóponttal négy csomópont csoportból áll, ha a `NumberOfReplicas` egyre van állítva, míg ha kettőre, akkor kettőből. Mgjegyzés: Minden csomópont csoportnak azonos számú csomópontból kell állnia.

Partíció: Ez egy része a klaszterben tárolt adatoknak. Annyi klaszter partíció van, ahány csomópont részt vesz a klaszterben. Minden csomópont felelős legalább egy másolatáért azon partícióknak, melyet hozzárendeltek (így legalább egy replika) elérhető a klaszterben. Egy replika egésze egyetlen csomóponthoz tartozik. Egy csomópont több replikációt is tud (és általában szokott is) tárolni.

Replika: Egy klaszter partíció egy másolata. Minden csomópont egy csoportban tárol egy replikát. A replikák száma megegyezik a csomópontok és csoportok hányadosával.

A telepítés alapjául a következő leírás szolgált:
http://dev.mysql.com/doc/refman/5.0/en/mysql-cluster-multi-computer.html.

```
server1 - adat csomópont
server2 - adat csomópont
server3 - adat csomópont és menedzsment csomópont
```

A klaszter összeállításának első lépése a menedzsment a szerverek konfigurálása. A konfiguráció három szinten zajlik. Egyrészt be kell állítsuk a menedzsment szervert. Ezt debian alatt a többi mysql konfigurációs adattól eltérően a /etc/myslq/ndb_mgmd.cnf fájlban tehetjük meg.

```
[NDBD DEFAULT]
NoOfReplicas=2
DataMemory=10MB
IndexMemory=25MB
MaxNoOfTables=256
MaxNoOfOrderedIndexes=256
MaxNoOfUniqueHashIndexes=128
```

```
[NDBD]
Id=2
# the first NDB Data Node
HostName=10.1.3.202
DataDir= /var/lib/mysql-cluster

[NDBD]
Id=3
```

```
[NDB_MGMD]
Id=1
# the NDB Management Node
HostName=10.1.3.203

# the second NDB Data Node
HostName=10.1.3.201
DataDir=/var/lib/mysql-cluster

[MYSQLD]
Id=4
# the first SQL node
HostName=10.1.3.203
```

Mint látható, itt nem csak a menedzsment szerver beállításait végezzük el, hanem magára a klaszterre vonatkozó konfigurációs adatokat is itt kell megadjuk. Ez azért van, mert a menedzsment szerver fogja biztosítani a beállítások elosztását a klaszterben. Miután a klaszter elindult, a konfigurációs adatok elosztásra kerülnek, és a klaszter működőképes akár a menedzsment csomópont (ok) jelenléte nélkül is. A következő lépés hogy, beállítsuk az adat csomópontokat is. Ezt ugyanazon konfigurációs állományban tehetjük meg, mint ahol a mysql szerver konfiguráció adatai találhatóak. (/etc/mysql/my.cnf)

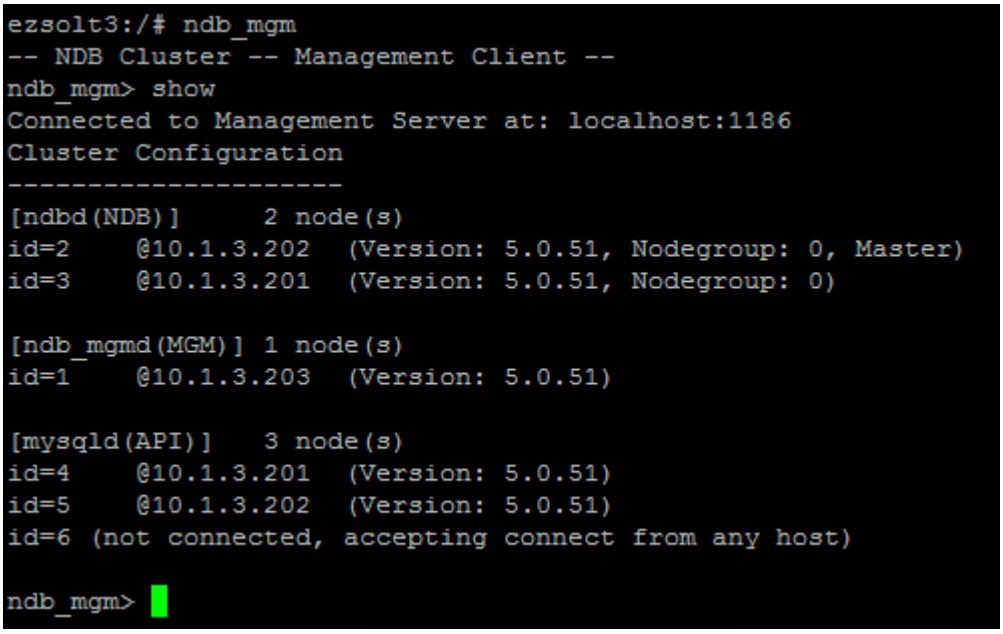
```
[MYSQL_CLUSTER]
ndb-connectstring=10.1.3.203
```

Az utolsó lépés pedig az SQL csomópontok konfigurálása, mely szintén a mysql szerver konfigurációs dájljában végezhető el.

```
[mysqld]
ndbcluster                      # run NDB storage engine
ndb-connectstring=10.1.3.203    # location of management
```

6.kép

MySQL Cluster működés közben



Forrás: saját szerkesztés (2009)

A klaszter indítását a menedzsment csomóponttal érdemes kezdeni. Ezt a /etc/init.d/mysql-ndb-mgm start paranccsal tehetjük meg. Majd pedig indítsuk el az egyes adat csomópontokat is a /etc/init.d/mysql-ndb start-initial paranccsal. A folyamatot a menedzsment szerveren követhetjük nyomon az ndb_mgm parancs használatával. A parancs kiadásával egy interaktív parancssort kapunk, mely segítségével menedzselhető adatbázis klaszterünk. Az aktuális állapot lekérdezése a show parancs segítségével érhető el. Fontos tudni, hogy ahhoz, hogy új csomópontot adjunk hozzá klaszterünkhöz, újra kell azt indítsuk. Szerencsére az újraindítás megoldható úgy is, hogy az ne járjon a klaszterünk leállásával egyetlen másodpercig sem. Ezt gördített újraindításnak (Rolling Restart) hívjuk. Ennek során csakúgy, mint az új kalszterünk elindításakor első lépésben a menedzsment csomópont, majd egyesével az adat csomópontok és végül az SQL csomópontok kerülnek újraindításra. Ezen módszer segítségével a klaszterünk egyes komponensei anélkül indíthatóak újra, hogy maga a klaszter ne üzemelne. Ugyanezen módszert kell alkalmazzuk akkor is, ha a klaszterünk szoftverfrissítésen vagy konfiguráció módosításon esik át. Az újraindítási műveletek egyébként bármely szerverről elvégezhetőek, mely kapcsolódhat a menedzsment szerverre, az ndb_mgm menedzsment kliensprogram segítségével.

Miután összeállt klaszterünk, akármely SQL csomópont segítségével létrehozhatunk ndbcluster típusú táblákat, melyek nem helyben, hanem a klaszteren kerülnek tárolásra. Minden módosítás helyesen jelenik meg, és ezen tábláinkat úgy kezelhetjük, mintha helyi táblák lennének. A konfiguráció során megadott rendundancia erejéig leállíthatjuk adat csomópontjainkat anélkül, hogy ez befolyásolná a klaszterünk működését. Ezzel tehát sikerült létrehoznunk egy magas rendelkezésre állású adatbázis klasztert.

6. Magas rendelkezésre állás biztosítása

Az eddig tárgyaltak során megvizsgáltuk, hogyan érhetjük el, hogy fájlrendszerünk, illetve adatbázisunk magas rendelkezésre állást biztosítson. Ez viszont még nem elegendő számunkra. Ha hiba következik be klaszterünk valamely csomópontján, fontos, hogy ezt kezelni tudjuk, és az adott csomópontról levegyük, eltereljük a terhelést, mert ha erre nem fordítunk figyelmet, fáradozásaink hasztalanok a felhasználók szemszögéből. Amennyiben egyedi kliens programmal dolgozunk, úgy van lehetőség a kérések elküldése előtt ellenőrizni, hogy a kiválasztott szerver jól működik-e, de bizonyos esetekben, például egy webkiszolgáló esetében nem járható út. A magas rendelkezésre állás biztosításához tehát szükségünk van egy további komponensre, mely vizsgálja a csomópontok állapotát, és hiba esetén automatikusan gondoskodik a szükséges lépésekről. Ha pedig egy korábban kiesett csomópontunk újra elérhetővé válik, akkor a csomópont újra beállításának munkáját automatizálja. Ezen feladatok megoldására született a Heartbeat programcsomag. A következőkben ezen programcsomag lehetőségeit vizsgáljuk meg.

6.1. Heartbeat, Pacemaker

A Heartbeat egy olyan démon, amely klaszter infrastruktúrát biztosít (kommunikációt és tagság kezelést) a csomópontok számára. Ez lehetővé teszi a csomópontok számára, hogy tudjanak a többi csomópont processzeinek jelenlétéről (vagy hiányáról), és könnyen tudjanak üzeneteket váltani egymás közt. Ahhoz, hogy a felhasználók számára könnyen használható legyen, a Heatrbeat démont egy klaszter erőforrás menedzserrel (CRM - cluster resource manager) kell kombináljuk, mely feladata a szolgáltatások indítása és leállítása a csomópontokon (IP cím, web szerver, stb.), mely lehetővé teszi a klaszter számára a magas rendelkezésre állást.[19]

A heartbeat egy egyszerű erőforrás kezelőt biztosít, mely a haresources névre hallgat, bár ez csupán két csomópont kezelését nyújtja, és nem ismeri fel az erőforrás szintű hibákat.

[19] A következő internetes forrás alapján, saját fordítás: http://www.linux-ha.org/, letöltve: 2009.10.25.

A Heartbeat 2.0.0-tól elérhető egy új erőforrás kezelő, mely megoldást nyújt a fentebbi problémákra. Az erőforrás kezelő 2007 ben kivált, és Pacemaker néven folytatta útját, hogy jobb támogatást tudjon nyújtani egyéb klaszter megoldásokra (mint amilyen az OpenAIS), és többé nem része a Linux-HA projektnek.

A pacemaker megjelenése előtt a heartbeat csupán két gép esetén tudott megoldásokkal szolgálni. Ez együtt járt azzal is, hogy a konfigurációja egyszerűbb, egyértelműbb volt. A pacemaker bevezetésével a konfigurációk csakúgy, mint a mysql klaszter esetében elosztva kerülnek tárolásra. A haresources re csupán egy példát mutatok, a telepítést és alkalmazást az új verzióval vizsgálom meg.

```
#node1  10.0.0.170 Filesystem::/dev/sda1::/data1::ext2
```

A fentebbi egy példa a haresources konfigurációjára. A beállítások nem mondhatóak rugalmasnak, mivel csupán egymás után felsorolt szolgáltatások adhatóak meg. A felsorolt szolgáltatásokat a heartbeat egymás után indítja el, illetve a leállításuk visszafelé történik. Ennél többet viszont nem tudunk megadni a szolgáltatások függőségét illetően. Ennél jóval szofisztikáltabb megoldást nyújt a Pacemaker.

A heartbeat2 telepítése debian rendszer alatt az aptitude install heartbeat2 paranccsal végezhető el. A heartbeat működéséhez három konfigurációs állományt kell feltétlenül beállítanunk. Az egyik a ha.cf, mely magára a heartbeatre mint szolgáltatásra jellemző opciókat fogad, a másik a haresources fájl, melyben a hagyományos, beépített crm használata esetén az erőforrások indítási sorrendje határozható meg . Ha a pacemakert szeretnénk használni, akkor pedig csupán a ha.cf fájban kell jelezzük, hogy extra crm et használnunk, a 'crm yes' konfigurációs opcióval. A harmadik fontos állomány, melyet létre kell hozzunk a /etc/ha.d/authkeys címen érhető el. Ebben adhajuk meg, hogy milyen módon szeretnénk azonosítani az egy klaszterben található heartbeat példnyokat, és mi az a közös titok (shared key), mely lehetővé teszi a kommunikáció titkosítását. Jelen esetben, az egyszerűség kedvéért egy egyszerű szöveges kulcsot adtam meg, mert a teszt környezetünkben nem fenyegeti veszély az adatokat, de valós körülmények között itt lehetőségünk van erősebb titkosítás beállítására is.

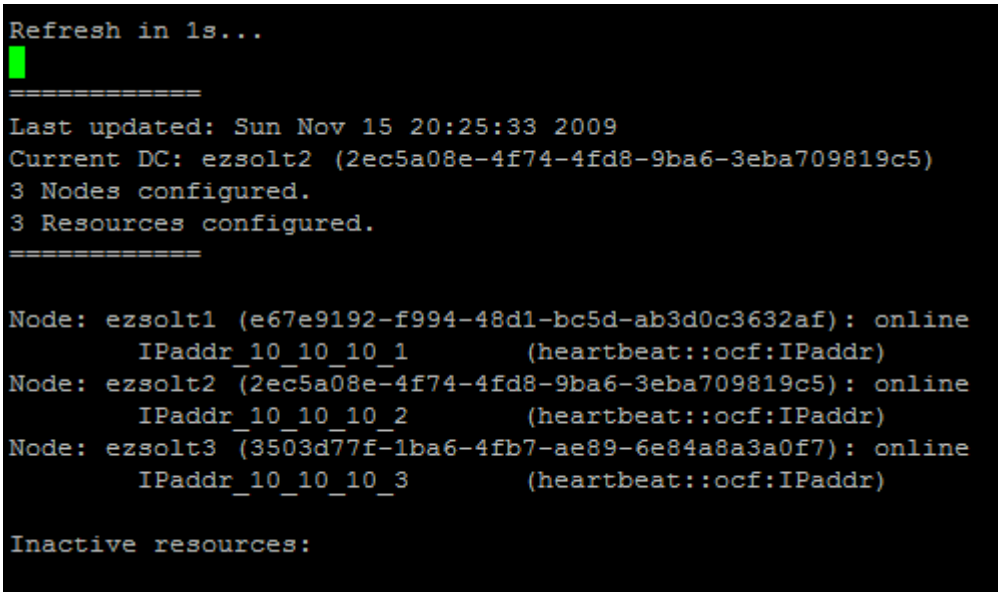
Mint azt korábban említettem, a ha.cf állományban a heartbeatre vonatkozó beállítások adhatóak meg. Itt kell tehát felsoroljuk azt is, hogy mely csomópontok (node) tagjai klaszterünknek. E mellett meg kell még adjuk, hogy mely interfészeket használhatja a heartbeat további csomópontok keresésére, és a kommunikációra. Ezt a bcast eth0 direktívával állítottam be. Itt nem csak ethernet interfészek megadására van lehetőség, hanem akár soros vonalon keresztül is beállítható a kommunikáció. Ez egyébként elég gyakran alkalmazott módszer, annak biztosítására, hogy ne jelentsen hibázási pontot az ethernet interfész. Mivel a heartbeat csomagok nem igényelnek nagy sávszélességet, így a soros vonali interfészek is megfelelőek a kommunikációra, és az ethernet interfész emghibásodásáról így értesíteni tudja az adott csomópont a többi résztvevőt.

```
node    pegazus2
node    pegazus3
node    pegazus4
```

Miután mindhárom szerveren elindítottuk a heartbeat szolgáltatást (/etc/init.d/heartbeat start) a klaszterünk elinudlt, de még igazából semmilyen hasznos szolgáltatást nem tud nyújtani. A konfigurációt mint korábban említettem, elosztva tárolja a pacemaker, és annak szerkesztését, módosítását a cibadmin segédprogramon keresztül szabályozhatjuk. Az aktuális állapot lekérdezhető a `crm_mon -i1 -nr` parancs segítségével, mely másodpercenként frissítve kijelzi, hogy mely erőforrás mely csomóponton talalható jelenleg, illetve azt is, hogy mely erőforrások állnak jelenleg, mert nem lehetséges a migrálásuk.

7.kép

A Heartbeat 2.0 és a Pacemaker működés közben



Forrás: saját szerkesztés (2009)

A konfiguráció frissítése a cibadmin paranccsal érhető el. Érdemes elsőként egy fájlba kimenteni a jelenlegi konfigurációt (`cibadmin -Q > cib.xml`), majd ezen fájlon elvégezni a megfelelő módosításokat, végül pedig a `cibadmin -Rx ./cib.xml` parancs segítségével felülírva a jelenlegi konfigurációkat, módosíthatjuk a beállításokat. Sajnos a jelenleg nem áll rendelkezésre webes vagy karakteres felületű konfigurációs interfész, a beállítások szerkesztése igen nehézkes.

8.kép

A Pacemaker IP cím hordozási XML konfigurációja

```
<constraints>
        <rsc_location id="rsc_location_group_1" rsc="group_1">
                <rule id="prefered_location_group_1" score="100">
                        <expression attribute="#uname" id="prefered_location_group_1_expr" operation="eq" value="ezsolt1"/>
                </rule>
        </rsc_location>
        <rsc_location id="rsc_location_group_2" rsc="group_2">
                <rule id="prefered_location_group_2" score="100">
                        <expression attribute="#uname" id="prefered_location_group_2_expr" operation="eq" value="ezsolt2"/>
                </rule>
        </rsc_location>
        <rsc_location id="rsc_location_group_3" rsc="group_3">
                <rule id="prefered_location_group_3" score="100">
                        <expression attribute="#uname" id="prefered_location_group_3_expr" operation="eq" value="ezsolt3"/>
                </rule>
        </rsc_location>
</constraints>
```

Forrás: saját szerkesztés (2009)

A tesztrendszeren sikerült beállítani az IP címek heartbeat által ellenőrzött működését. Ehhez a meglevő IP címek mellé, az eth0 interfészre került egy-egy virtuális interfész kiosztásra a 10.10.10.* C osztályú IP tartományban egy-egy cím. A címeket a heartbeat kezeli. A konfigurációban megadott szabályok alapján mindhárom virtuális géphez hozzá lett rendelve egy-egy alapértelmezett cím megfelelően nagy értékű súllyal. Ez biztosítja, hogy normális működés esetén mindhárom gép a saját IP címét használja. Ha pedig valamely gép elérhetetlenné válik, a heartbeat átmozgatja az erőforrást egy másik gépre. Az elosztás jelenleg véletlenszerű, de szabályok segítségével megadhatóak prioritások is, hogy mely IP címek mely gépekre kerüljenek, vagy kerülhetnek.

7. Összefoglalás

Dolgozatom végéhez érve, azt kell mondjam, kitűzött céljaim sikerült elérni. Az elosztott adattárolás és a magas rendelkezésre állás már nem csak a külön, erre a célra fejlesztett alkalmazások és a nagyvállalatok kiváltsága. A feladat szabad forrású és ingyenes eszközökkel is kiválóan elvégezhető, ha alkalmazásuk nem is annyira kényelmes, felépítésük és működésük tökéletesen alkalmas egy skálázható magas rendelkezésre állású rendszer kialakítására. Az adattárolásra véleményem szerint a glusterfs a legalkalmasabb eszköz jelenleg, mert igen rugalmas, és lehetőséget biztosít a céljainknak megfelelő rendszer felépítésére. Az egyetlen hátulütője, hogy még viszonylag fiatal szoftver, így még nem rendelkezik széleskörű támogatottsággal. Adatbázis szinten pedig a mysql igen komoly megoldást nyújt klaszter technológiája révén. Mindezen komponenseket pedig a heartbeat 2.0 verziója kitűnően összefogja, és biztosítja az egyéb kapcsolódó erőforrások kezelését is.

Dolgozatom amellett, hogy bemutatta és végigvezette az olvasót a szükséges szoftverkomponensek telepítésén, melléktermékként létrehozott egy olyan, három szerverből álló tesztkörnyezetet, mely bármikor alkalmazható, akár egyetlen gépen szeretnénk kipróbálni a leírtakat, akár külön szerverekre szeretnénk telepíteni azt, hogy megvizsgáljuk, egy adott rendszer lehetőségeinek mely kombinációk felelnek meg a legjobban. Bízom benne, hogy olvasóim haszonnal forgatják dolgozatomat, és sikerült megoldásokat nyújtanom a felmerült problémákra.

8. Hivatkozásjegyzék

- Crichlow, J. M. (2003): Elosztott rendszerek. Kiskapu Kft., Budapest

- Distributed file system
  http://en.wikipedia.org/wiki/Distributed_file_system, letöltve: 2009.10.26.

- Authentikáció
  http://fogalomtar.eski.hu/index.php/Authentikáció, letöltve: 2009.10.11.

- Wikipedia: File Locking
  http://en.wikipedia.org/wiki/File_locking, letöltve: 2009.10.18.

- Wikipedia: Gyorsítótár
  http://hu.wikipedia.org/wiki/Gyorstár, letöltve: 2009.10.18.

- WinPcap főoldal
  http://www.winpcap.org/default.htm, letöltve: 2009.10.19.

- Az NFS és mások összehasonlítása (Compersion of NFS vs. others)
  http://wiki.linux-nfs.org/wiki/index.php/Comparison_of_NFS_vs._others#NFSv3

- Wikipedia: Samba
  http://hu.wikipedia.org/wiki/Samba, letöltve: 2009.10.19.

- Samba Howto – Chapter 17. File and Record Locking
  http://www.samba.org/samba/docs/man/Samba-HOWTO-Collection/locking.html, letöltve: 2009.10.19.

- OpenAfs dokumentáció: Felhasználói kézikönyv
  http://docs.openafs.org/UserGuide/ch01.html#HDRWQ3, letöltve: 2009.10.19.

- Wikipedia: Global File System
  http://en.wikipedia.org/wiki/Global_File_System, letöltve: 2009.10.16.

- OpenAFS Windows kiadási ismertető
  http://docs.openafs.org/ReleaseNotesWindows/ch03s19.html, letöltve: 2009.10.25.

- Wikipedia: Fájlrendszerek listája
  http://hu.wikipedia.org/wiki/Fájlrendszerek_listája#ElosztottPárhuzamosFájlrendszerek, letöltve: 2009.10.28.

- Glusterfs leírás
  http://www.gluster.com/products/index.php, letöltve: 2009.10.28.

- Glusterfs dokumentáció: glusterfs bemutatása
  http://www.gluster.com/community/documentation/index.php/GlusterFS_Introduction#Overview, letöltve: 2009.10.21.

- Wikipedia: Replication
  http://en.wikipedia.org/wiki/Replication_%28computer_science%29, letöltve: 2009.10.28.

- Wikipedia: Peet-to-peer fogalom
  http://hu.wikipedia.org/wiki/Peer-to-peer, letöltve: 2009.10.18.

- MySQL online referencia kézikönyv
  http://dev.mysql.com/doc/refman/5.0/en/images/multi-comp-1.png, letöltve: 2009.10.28.

- Wikipedia: MySQL Cluster fogalom
  http://en.wikipedia.org/wiki/MySQL_Cluster, letöltve: 2009.10.28.

- MySQL referencia kézikönyv: MySQL klaszter csomópont csoportok
  http://dev.mysql.com/doc/refman/5.0/en/mysql-cluster-nodes-groups.html.

- MySQL referencia kézikönyv: Többgépes mysql cluster http://dev.mysql.com/doc/refman/5.0/en/mysql-cluster-multi-computer.html.

- Linux-ha nyitólap, leírás (ha – High Availability – Magas Rendelkezésreállás) http://www.linux-ha.org/, letöltve: 2009.10.25.