Szolgáltatásbiztonság vizsgálata

Modellezési feladatok

Tantárgy:	Intelligens rendszerfelügyelet (VIMIA370)
Szerkesztette:	Micskei Zoltán
Készítették:	Horváth Gergely J., Kocsis Imre, Micskei Zoltán, Szatmári Zoltán, Szloboda István, Tóth Dániel
Utolsó módosítás:	2014. május 2., verzió: 1.6

Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

1 Bevezető

Az *Intelligens rendszerfelügyelet* tantárgy keretében különböző rendszerek modellezési lehetőségeibe is belekóstolunk. Akár egy egyszerű modell elkészítése is nagyon hasznos lehet:

- segít összegyűjteni és megérteni az adott szakterület fogalmait,
- szisztematikus módszert ad, hogy összegyűjtsük az előkerülő fogalmakhoz kapcsolódó szabályokat és kényszereket („egy rendeléshez hány kapcsolattartót lehet megadni?", „kötelező-e kitölteni az értesítési telefonszámot, ha az e-mail meg van adva?", stb.),
- szabványos modellezési nyelvek használatával egyértelműbbé tehetjük, hogy mit értünk az egyes elemeken és kapcsolatokon,
- lehetőség nyílik, hogy automatikus ellenőrzéseket valósítsunk később meg (megadtunk-e minden szükséges adatot, kiszámoljuk a rendszer bizonyos jellemzőjét).

A tantárgy keretében két különböző feladatot néztünk meg részletesebben a félév során:

- *Adatmodellek készítése*: egy adott terület fogalmait és azok kapcsolatát gyűjtjük össze. Tipikusan ez egy magas szintű, kezdeti modell, ami még nem az implementáció közeli részletekre koncentrál.
- *Szolgáltatásbiztonság vizsgálata*: összetett rendszerek rendelkezésre állását, hibatűrését, adott hibajelenségek diagnosztikáját segítjük különböző hibamodellek összeállításával.

A tantárgy vizsgájának gyakorlati részében ilyen feladatok megoldását várjuk el, ez a segédlet a vizsgára való felkészülést segíti. Javasoljuk, hogy a kidolgozott mintapéldákat is először mindenki próbálja *önállóan* megoldani, és csak utána nézze meg a megoldást. A modellezés is egy olyan készség, amit csak gyakorlással lehet fejleszteni, ezért érdemes utána a gyakorló feladatokat is önállóan megoldani (pusztán a megoldás átolvasása még nem elég ahhoz, hogy később alkalmazni is tudjuk az ott látott ismereteket).

1.1 Szolgáltatásbiztonság, hibatűrés modellezése, diagnosztika

A *szolgáltatásbiztonság* egy összefoglaló fogalom, a rendszer azon képessége, hogy igazoltan bízni lehet a szolgáltatásában. Ez sok mindent foglalhat magában, mennyire képes a lehetséges hibákat lekezelni, milyen teljesítményt nyújt nagy terhelés ellen, mennyire biztonságos, stb. Egy konkrét összetett rendszer, ami hardver és szoftverkomponenseket is tartalmaz, valamint megjelenik benne az emberi tényező is, szolgáltatásbiztonsági jellemzőit meghatározni bonyolult feladat. A tantárgy keretében megnéztük pár egyszerűbb módszert hibamódok és kapcsolataik összegyűjtésére, amit már fel lehet használni arra, hogy egy adott hibajelenség okának a felderítésében el tudjunk rendszerezetten indulni.

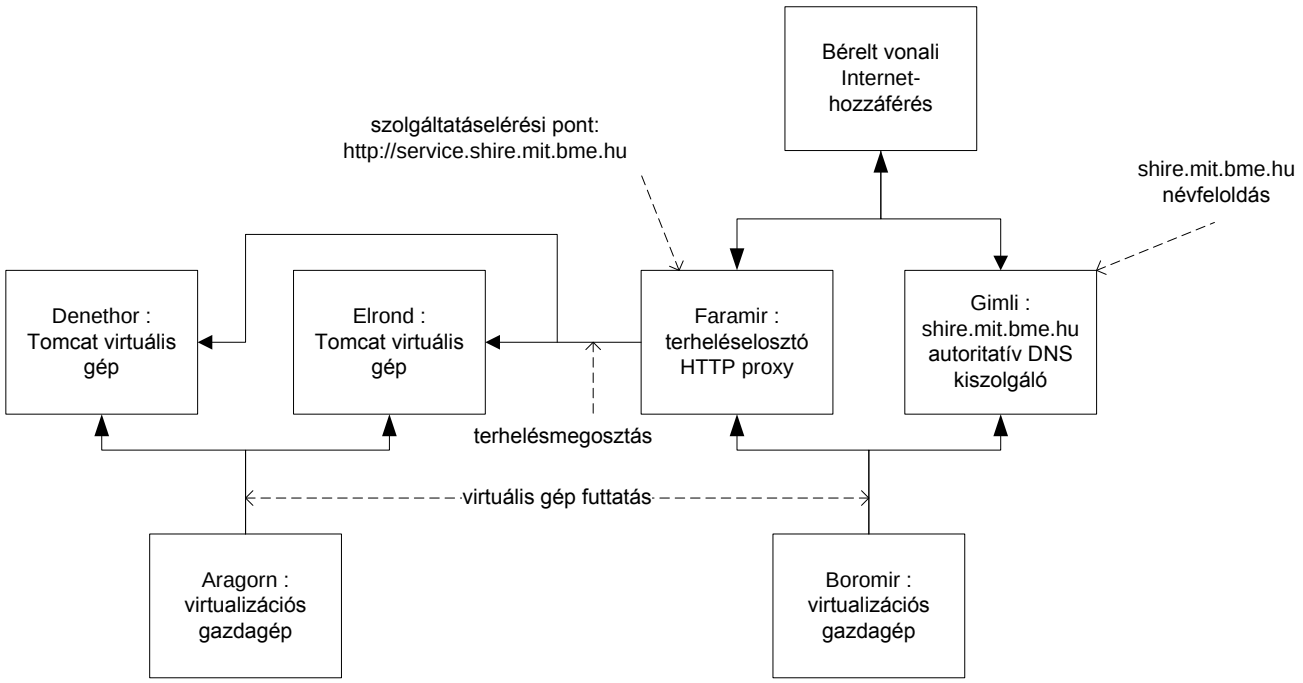
A felkészüléshez nézzük át a szolgáltatásbiztonság alapjaival foglalkozó előadást (hibák típusai, terjedésük módjai, hibalisták és hibafák). Ezen kívül tisztában kell lenni a tantárgyban előkerülő módszerekkel és technológiákkal (pl. egy terheléselosztó fürtöt használó webes alkalmazás hibájának elhárításához tisztában kell lennünk azzal, hogy hogyan is működik egy ilyen fürt).

2 Kidolgozott mintapéldék

2.1 Terheléselosztó webes rendszer

2.1.1 A feladat szövege

A mellékelt ábrán egy egyszerű dinamikus webszolgáltatást megvalósító főbb komponensek és azok kapcsolatai láthatóak. A rendszer az Internethez egy bérelt vonali szolgáltatáson keresztül kapcsolódik. A rendszer tartalmaz egy DNS kiszolgálót (shire.mit.bme.hu), mely többek között a webszolgáltatás elérési pontjának névfeloldásáért is felel a kliensek felé. A szolgáltatáselérési pont egy egyszerű HTTP proxy, mely round-robin terhelésmegosztást valósít meg két Tomcat alkalmazás-kiszolgáló között. A terhelésmegosztás mellett a proxy egyben hibatűrő működést is biztosít; egy alkalmazás-kiszolgáló kiesésesének érzékelése esetén a nem teljesített kérés(eke)t megismétli a másik kiszolgáló irányába. Ő maga HTTP szintű hibát csak akkor generál, ha egyik kiszolgáló sem válaszol a kérésekre. Minden említett kiszolgáló virtuális gépként került megvalósításra, két gazdakiszolgáló között szétosztva. A helyi hálózat hibáit a feladat megoldása során elhanyagolhatja.



a) Készítsen hibafát a következő eseményhez: a kliensek böngészőjében lelassul a web-szolgáltatás oldalainak megjelenítése, bár a szolgáltatás terhelése nem változott. (3 pont)

b) Készítsen hibafát a következő eseményhez: a kliensek böngészőjében „nem megy az oldal” (feltételezheti, hogy a névfeloldással nincs probléma). (4 pont)

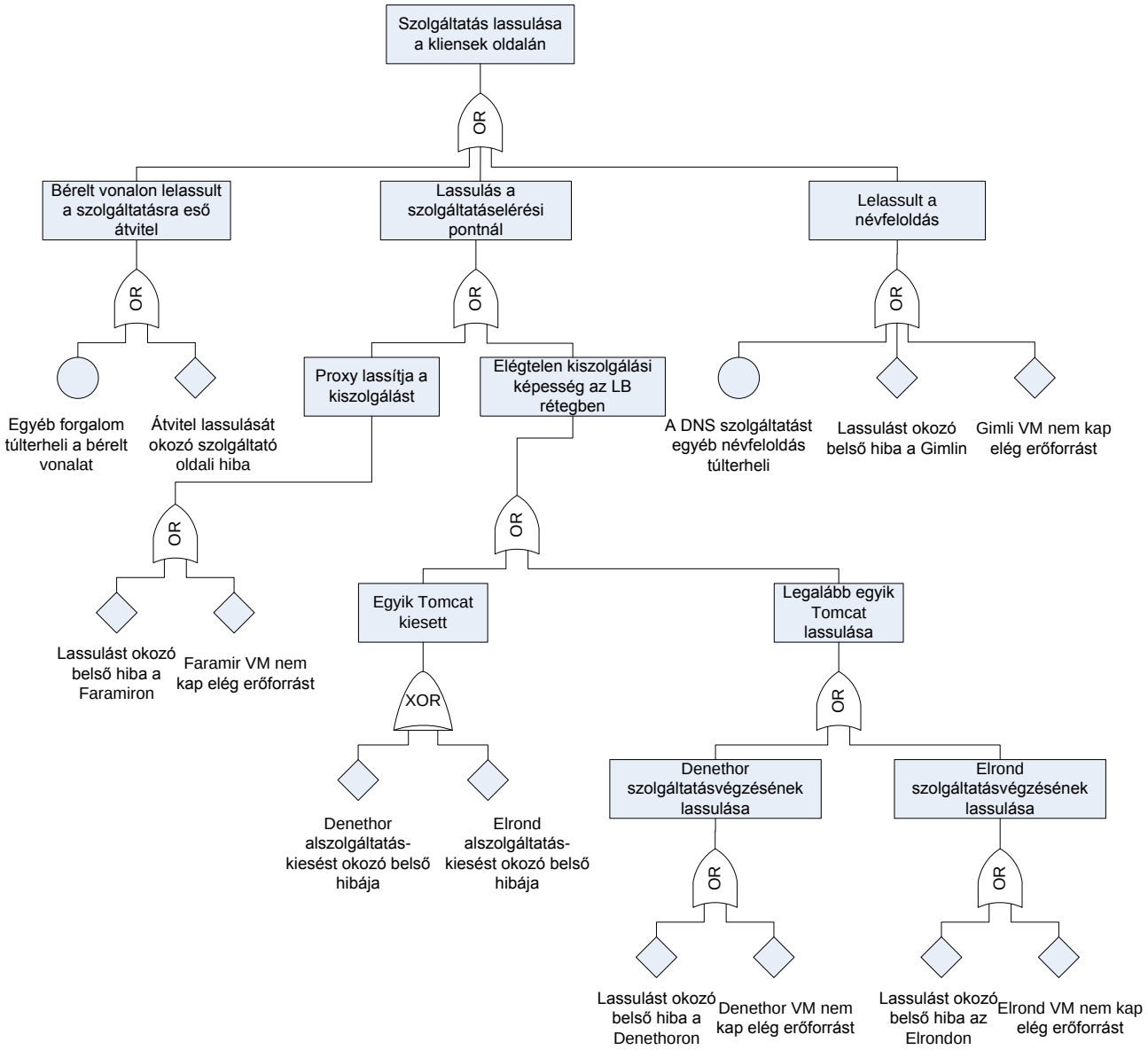
c) Jelenleg a két Tomcat kiszolgáló ugyanazon a gazdagépen található. Növekszik-e a szolgáltatás megbízhatósága, ha az egyik Tomcat kiszolgálót átmozgatjuk egy új, más gépet nem futtató gazdagépre? Válaszát indokolja formálisan is; egyszerűsítésképpen valamely komponens „a misszióidő alatti elromlását” modellezheti egyszerű valószínűségi eseményként mely valamely – ismeretlen – valószínűséggel következik be. (Tehát sem a misszióidőnek, sem pedig a szokásos valószínűségi sűrűségeloszlás-függvénynek nem kell megjelennie.) (3 pont)

Megoldásaiban ügyeljen az áttekinthetőségre, a szakmai pontosságra és arra, hogy a kísérő szöveg megfelelő részletességű és mélységű legyen!

2.1.2 Egy lehetséges megoldás

Az 1. ábra az a) feladat egy meglehetősen részletes megoldása. Figyeljük meg a különböző alkalmazott szintaktikai elemeket.

- Téglalap: „köztes esemény”. Két logikai kaput nem kötünk közvetlenül össze; egy kapu kimenetén az adódó kompozit eseményt illik szerepeltetni (a hibafa szerkesztő eszközök két kapu közvetlen összekötését nem is nagyon engedik meg). Egyébiránt mivel a hibafákat „top-down” építjük, a köztes események természetesen adódnak.
- Kör: alapesemény (*basic event*). Erről előadáson volt szó.
- Logikai kapuk: ezek a szokásosak; a 2010. I. vizsgán az OR és AND mellett szükség volt XOR-ra is, de ez kifejezhető AND, OR és negálás (akár események jelentése szintjén) segítségével is, mint arra sokan helyesen rá is jöttek.



1. ábra Egy megoldás az a) feladatrészre

Az ábrán látható még „gyémánt” alakú levél elem is; ezzel a „tovább ki nem bontott” eseményeket (*undeveloped event*) szokás kifejezni. Létezik még egy harmadik levél-típus is, a „házikó” alakú (⌂), ezzel a rendszer szemszögéből nézve külső eseményeket fejezzük ki szokásosan.

A vizsgán tipikusan szintaktikai hibákért, egy-két eset kivételével, **nem vontunk le pontot**; ilyenek például a köztes események elhagyása, ki nem bontott esemény helyett alapesemény szerepeltetése, stb. (Persze ha logikai kapuk egyáltalán nem szerepelnek a megoldásként beadott ábrán, az már más kérdés…)

A következő típushibákkal találkoztunk.

1. Az XOR kapu helyett egyszerű OR szerepeltetése – ha a Denethor és az Elrond egyszerre esik ki, akkor a szolgáltatás nem lelassulni fog, hanem kiesni.
2. Túlságosan is ötletszerű hibák és azok közötti összefüggések. A megoldásban elvárt ábrának a rendelkezésre álló idő rövidsége miatt nem kellett ennyire részletesnek lennie (más kérdés, hogy a kibontatlan események mentén ezt a hibafát is lehetne még tovább bontani, illetve itt-ott egy-két elhanyagolt eseményosztály azért fellelhető). Viszont az is látszik, hogy a hibafa top-down megközelítésű felvétele során törekedni kell arra, hogy szintenként az összes reálisan szóba jövő lehetőséget lefedjük.
3. A 2. pontban írtak folytatásának tekinthető az, hogy az események megnevezésének szükségesen és elégségesen precíznek kell(ene) lennie. Pl. az, hogy „Faramir hiba” ebben az esetben azért nem jó, mert a Faramirnak létezik olyan hibamódja, ahol egyáltalán nem nyújt szolgáltatást. Fordított irányban: felesleges elkezdeni a különböző „félrekonfigurációs” hibaokokat felsorolni, hiszen a lista semmiképp sem lesz teljes, és így (a hibafa implicit „zárt világ” logikája miatt) kimaradnak olyan események, melyeknek szerepelniük kellene. Ilyenkor az a helyes megközelítés, hogy egy tovább ki nem bontott (vagy akár alap-) eseményt veszünk fel, mely nem a *hibaokot*, hanem a lokális *hibahatást* ragadja meg – pl. „helytelen konfiguráció okozta szolgáltatáskiesés”.

b) feladat

Ugyanazok vonatkoznak rá, mint az a) feladatra.

c) feladat

Tegyük fel, hogy a Tomcat virtuális gépek kieséséhez vezető belső hibák valószínűsége P_t; tegyük fel továbbá, hogy a virtualizációs hosztok azonosak, kiesésük valószínűsége pedig: P_v. Látható, hogy a Faramirt már nem érinti az átalakítás és egyik más komponens sincs a Tomcat-ekkel és az őket futtató hoszt(okk)al közvetlen kapcsolatban. Ennek megfelelően (és tudva, hogy a külvilág felé vállalt szolgáltatás mindenképpen megszűnik, ha a terheléselosztott rétegben kiesik a szolgáltatás) elég, ha csak a terheléselosztott réteg szolgáltatásával foglalkozunk. Próbáljuk meg a megbízhatóságot („annak a valószínűsége, hogy nem romlik el a misszióidő alatt”) felírni erre a szolgáltatásra mindkét esetben!

Az első esetben a szolgáltatás akkor működik, ha sem a hoszt nem esik ki, sem pedig egyszerre a két Tomcat:

$$P_1 = (1 - P_v)(1 - P_t^2)$$

A második esetben a szolgáltatás akkor működik, ha nem esik ki mind a két gép + Tomcat páros; egy páros pedig akkor nem esik ki, ha sem a Tomcat, sem a hoszt nem esik ki:

$$P_2 = 1 - \left(1 - (1 - P_v)(1 - P_t)\right)^2$$

Rendezzük át a második kifejezést!

$$P_2 = 1 - 1 + 2(1 - P_v)(1 - P_t) - (1 - P_v)^2(1 - P_t)^2$$

Sejtésünk szerint képezzünk egy egyenlőtlenséget:

$$\begin{gathered} P_1 < P_2 \\ (1 - P_v)(1 - P_t^2) < 2(1 - P_v)(1 - P_t) - (1 - P_v)^2(1 - P_t)^2 \\ (1 - P_t^2) < 2(1 - P_t) - (1 - P_v)(1 - P_t)^2 \\ 1 - P_t^2 < 2 - 2P_t - (1 - P_v)(1 - 2P_t + P_t^2) \\ -P_t^2 < 1 - 2P_t - 1 + 2P_t - P_t^2 + P_\mathrm{v} - 2\mathrm{P_v P_t} + \mathrm{P_v P_t^2} \end{gathered}$$

$$0 < P_v - 2P_vP_t + P_vP_t^2$$

$$0 < 1 - 2P_t + P_t^2 = (1 - P_t)^2$$

Tehát feltételezve, hogy a valószínűségek nulla és egy közé esnek, valóban nő a megbízhatóság.

A vizsga javításánál elfogadtunk olyan megoldást is (1 pontot levonva), ami a Tomcat réteget nem vette figyelembe. Furcsamód rengeteg olyan próbálkozás volt valószínűségek számítására, ahol két valószínűség úgy lett összeadva, hogy azok értéke nagyobb lehetett, mint 0.5… (Azaz lehetőleg csak úgy ne adjunk össze két, eltérő valószínűségi változókhoz tartozó kimenet valószínűségét megadó változót, mert a végeredmény nagyobb lehet mint 1, ami rossz ómen a megoldás helyességével kapcsolatban.)

2.2 Kisvállalati infrastruktúra

2.2.1 A feladat szövege

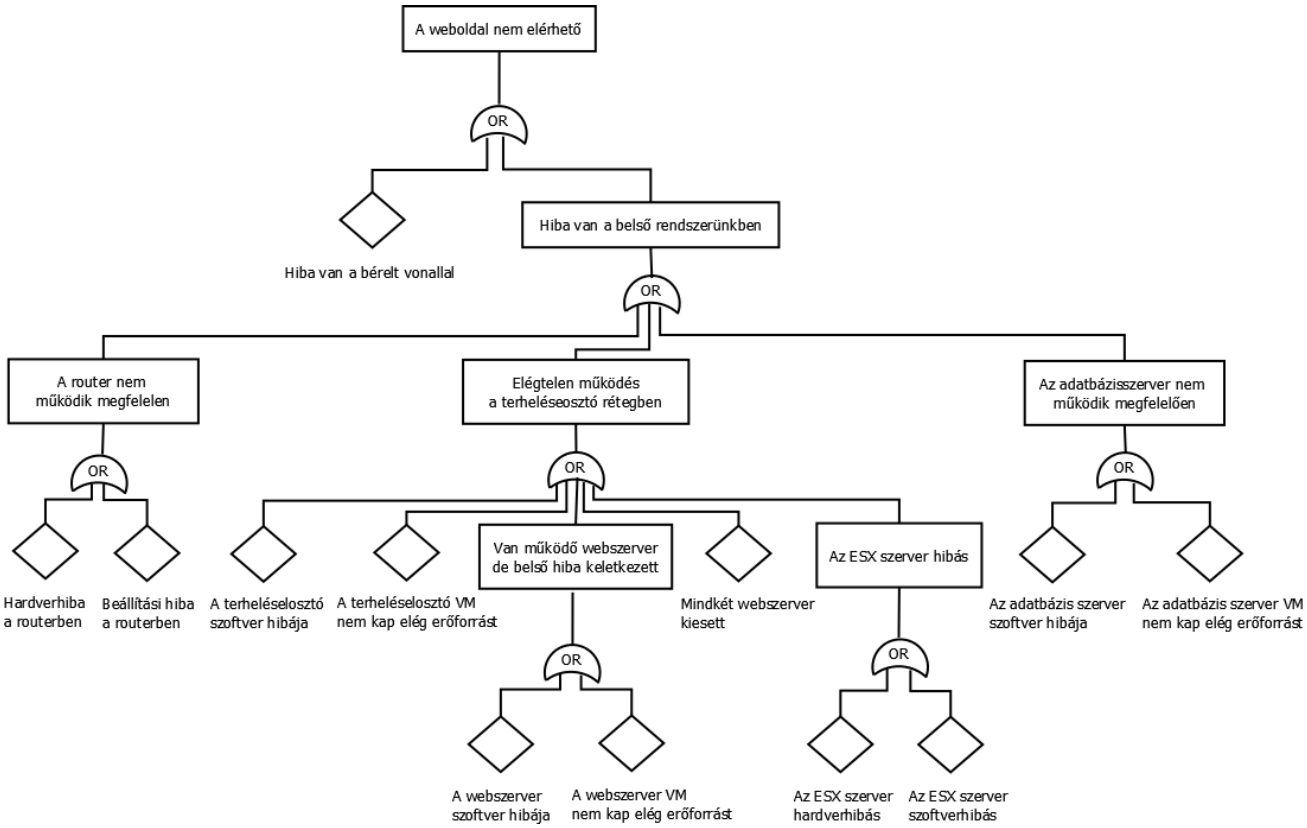
Egy vállalat saját infrastruktúrával szolgálja ki a weboldalát. A cég hálózata egy linuxos routeren keresztül kapcsolódik az Internet elérését biztosító bérelt vonalhoz, ez az eszköz NAT-ol a belső hálózat gépeinek. A weboldalt virtuális gépek szolgálják ki, melyek egy darab ESX szerveren futnak. Jelenleg két darab webszerver működik egy terheléselosztó fürtben, az adatokat pedig egy külön adatbázis szerver tárolja. A weboldal működéséhez szükséges az adatbázis működése is.

1. Gyűjtsön össze legalább 5 meghibásodást, amik meggátolhatják a weboldal működését, és ábrázolja ezeket egy hibafában! (10 pont)
2. Egy táblázatban javasoljon módszereket, hogy mivel lehetne védekezni a fenti meghibásodások ellen! (10 pont)

2.2.2 Egy lehetséges megoldás

1.) feladat:

Egy lehetséges részletes hibafát mutat be a következő ábra:



2.) feladat:

A javasolt védekezési módszerek a különböző meghibásodások ellen:

Meghibásodás jellege:	Védekezési javaslatok:
Hiba a bérelt vonallal	Szolgáltató váltás (magasabb rendelkezésre állású csomag)
Hardverhiba a routerben	Router duplikálása (redundancia) Jobb minőségű router vásárlása
Beállítási hiba a routerben	Figyelmes, többször ellenőrzött paraméter-beállítás
Terheléselosztó szoftverhiba	A terheléselosztó többszörözése Más terheléselosztó szoftver használata
A terheléselosztó VM nem kap elég erőforrást	Figyelmesebb ESX szerver konfiguráció (korlátok beállítása) Hardverbővítés
Webszerver szoftverhiba	Különböző webszerverek használata (SW redundancia)
A webszerver VM nem kap elég erőforrást	Figyelmesebb ESX szerver konfiguráció Hardverbővítés
Mindkét webszerver kiesése	Még egy példány használata
ESX szerver hardverhiba	Két fizikai gép beállítása ESX szervernek
ESX szerver szoftverhiba	Stabil, bejáratott ESX verzió használata
Adatbázis szerver szoftverhiba	Az adatbázis szerver többszörözése
Az adatbázis szerver VM nem kap elég erőforrást	Figyelmesebb ESX szerver konfiguráció Hardverbővítés

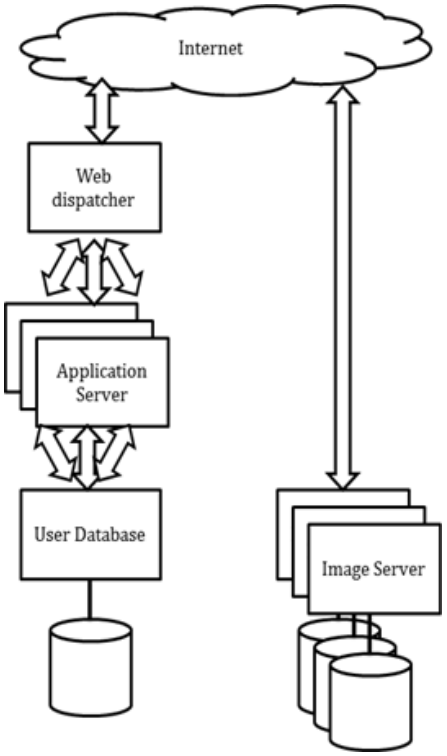
2.3 Web 2.0 közösségi oldal

2.3.1 A feladat szövege

Egy garázscég elindított egy – napjainkban rendkívül népszerű – Web 2.0 közösségi oldalt, mely tárolja a felhasználók adatait és kapcsolati hálóját, valamint a felhasználók képeket tölthetnek fel a profiljukra. Költségtakarékosságból egy olyan rendszert alakítottak ki, mely teljes egészében (Infrastructure as a Service típusú) cloud computing szolgáltatótól bérelt virtuális gépeken fut. A garázscég a terhelés függvényében rendelhet ideiglenesen új virtuális gépeket, illetve törölhet olyanokat, amik éppen nem szükségesek.

A cloud szolgáltató a következő garanciákat biztosítja:

- A virtuális gépek folyamatos futása nem garantált, ritkán előfordulhatnak kiesések (olcsó szolgáltatási szintű csomagra fizettek elő).
- Ha egy virtuális gép futása a szolgáltató hibájából megszakad, akkor a szolgáltató azonnal újraindítja azt egy másik hoszt gépén. Az újraindítás természetesen időbe kerül.
- A virtuális gépekhez külön díjért rendelhető perzisztens adattár („Volume"). Az adattár tartalmát a szolgáltató hibatűrő módon tárolja, és a virtuális gépek törlése után is megmarad a tartalma, mely később más virtuális géphez hozzárendelhető.



Ennek megfelelően a garázscég olyan architektúrát alakított ki, melyben a következő elemek vannak:

- Egy közös front-end webszerver, mely központi komponensként elosztja a beérkező kéréseket az alkalmazásszerverek felé. Nem tárol perzisztens adatot.
- Tetszőleges számú alkalmazásszerver, mely generálja a bejelentkezett felhasználók számára a web oldalakat, HTML hivatkozással a megfelelő kép szerveren tárolt képekre. Csak az éppen folyamatban lévő munkameneteket (session) tárolja.
- Egy közös adatbázisszerver, mely tárolja a felhasználók adatait és kapcsolatait. Az adatbázis perzisztens volume-on van.
- Tetszőleges számú kép szerver, mely kiszolgálja a felhasználók által feltöltött képeket. Mindegyikhez egy-egy perzisztens volume tartozik, a volume-ok tartalma eltérő, szét vannak osztva a képek közöttük.

Feladatok:

a) Rajzolja fel a kiszolgálók állapotgépét (elég egy általános kiszolgálóra megadni). Fontos megkülönböztetni a virtuális gépet és a rajta futó alkalmazást, mint elemi hibaforrást. Azt itt definiált hibaállapotokat használja később elemi hibákként! (2 pont)

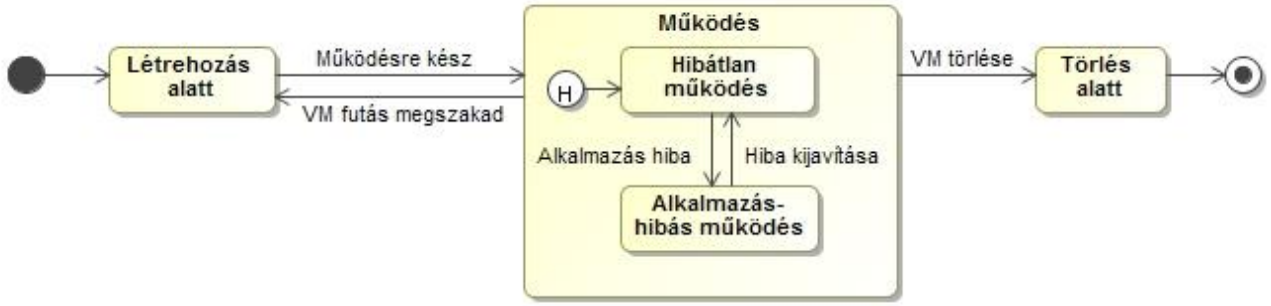
b) Néhány felhasználótól olyan hibabejelentés érkezik, hogy nem tudnak belépni, mindig visszakerülnek a login oldalra (természetesen konkrét hibaüzenet nélkül). Az oldal többszöri betöltésével a hibajelenség elmúlik. Készítsen hibafát erre az esetre! Feltételezhető, hogy a szolgáltató csak a fent említett módon hibázik, a hálózati kapcsolóelemek hibáival most nem kell foglalkozni. (3 pont)

c) A fenti hiba kijavítása után most néhány felhasználó arra kezdett panaszkodni, hogy a képeik sosem jelennek meg az oldalukon, hanem az oldal betöltése után azonnal a kis „törött kép" ikon jelenik meg a böngészőben. Más felhasználóknál viszont minden rendben van. Készítsen erre az esetre is hibafát! (3 pont)

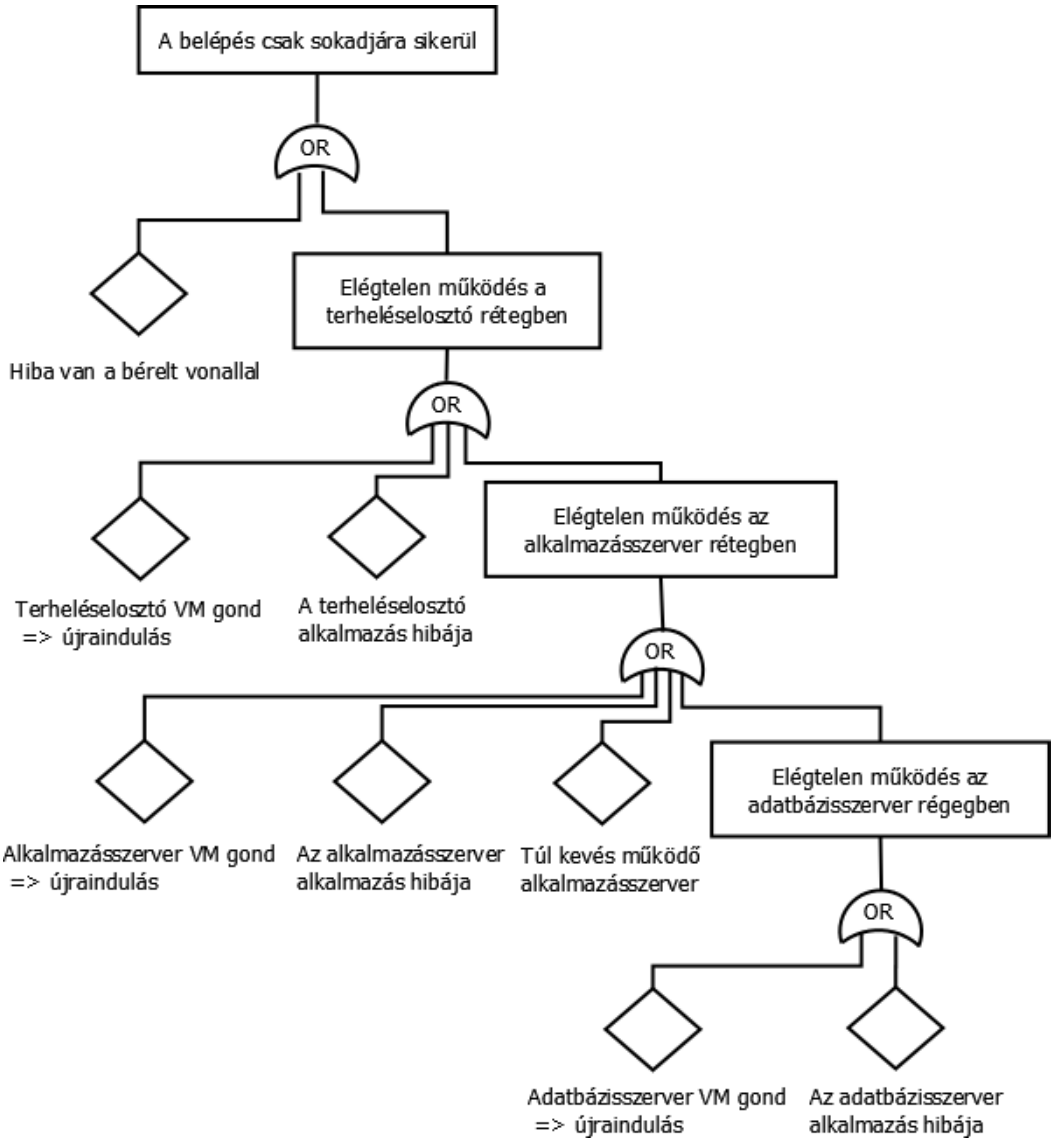
d) Az egyik rendszergazda valahogyan „megoldotta" a fenti problémát, de most olyan bejelentés érkezett, hogy az oldal bejön, de nagyon hosszú ideig töltődnek a képek és végül nem jelennek meg. Mit csinálhatott a rendszergazda? Az előző hibafán jelezze, hogy miben térhet el a mostani eset az előzőtől! Mi lehet az alapvető tervezési hibája a fenti rendszernek? (2 pont)

2.3.2 Egy lehetséges megoldás:

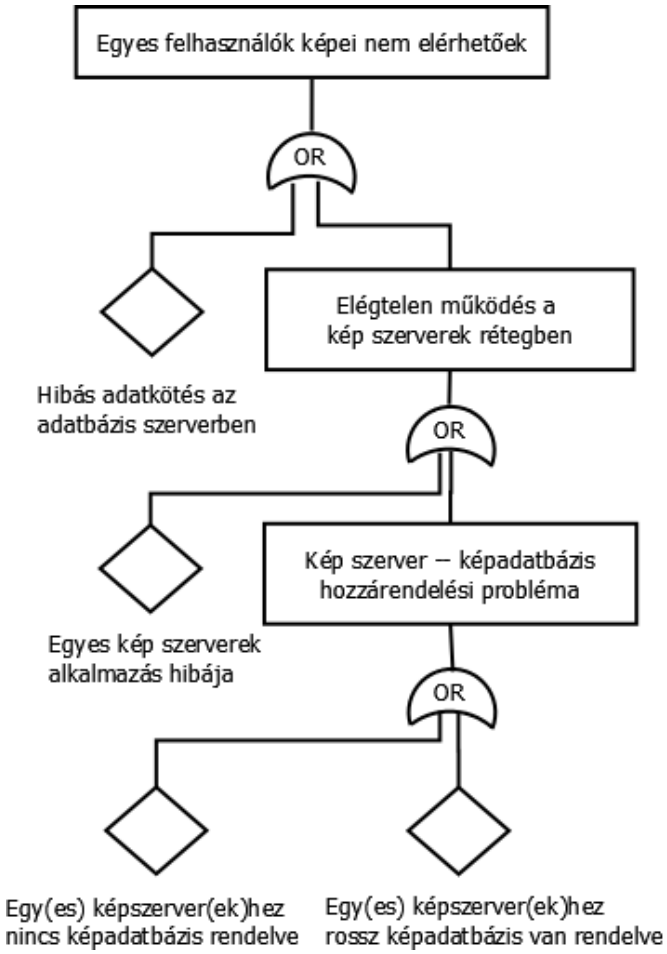
a.) feladat:



b.) feladat:



c.) feladat:



d.) feladat:

A rendszergazda valószínűleg vagy beállított minden kép szerverhez minden kép-adatbázist, vagy egy kép szervert hagyott meg, amelyhez minden kép adatbázist hozzárendelt; ez által az elosztottság előnyeit szüntette meg. Érthetően a nagy terhelést nem tudja kiszolgálni, és valószínűleg időtúllépések miatt nem jelenik meg végül a sok kép.

Tervezési hiba: ha már elosztott megoldásokban gondolkodunk, érdemes az adatbázist is redundánsan tárolni, a háttérben szinkronizálva, a gyorsabb elérés érdekében. Hasonlóképp a kép szervereknél is lehetne terhelés elosztás és egy adatbázishoz több kép szerver.

3 Korábbi vizsgafeladatok

Ebben a fejezetben korábbi vizsgafeladatokat gyűjtöttünk össze. Érdemes a frissebbek megoldásával kezdeni a gyakorlást, hasonlókra lehet számítani általában a vizsgákon.

3.1 Belépés címtáras környezetben (2009. 05. 27.)

Cégünk a következő belső infrastruktúrával rendelkezik. Tíz darab Windows XP-t futtató kliensgépünk van, amik egy 100 Mbit/sec-es switch-en keresztül csatlakoznak a szerverünkhöz. A szerveren Windows Server 2008 fut, valamint egy Active Directory címtár. A gépek be vannak léptetve az Active Directory tartományba, azaz a felhasználók hitelesítését az Active Directory címtár végzi el. A Windows viszont eltárolja az utolsó 3 sikeres tartományi belépést, így ha épp nem elérhető a címtár valamiért, de a felhasználó már belépett korábban erről a gépről, akkor a helyileg tárolt adatok segítségével el tudja végezni a kliens is a beléptetést.

a. Rajzoljon fel egy hibafát, ami azt ábrázolja, hogy milyen események vezethetnek ahhoz, hogy egy felhasználó nem tud belépni egy konkrét kliens gépről! A hibafának legalább 4 elemi eseményt kell tartalmaznia! (8 pont)

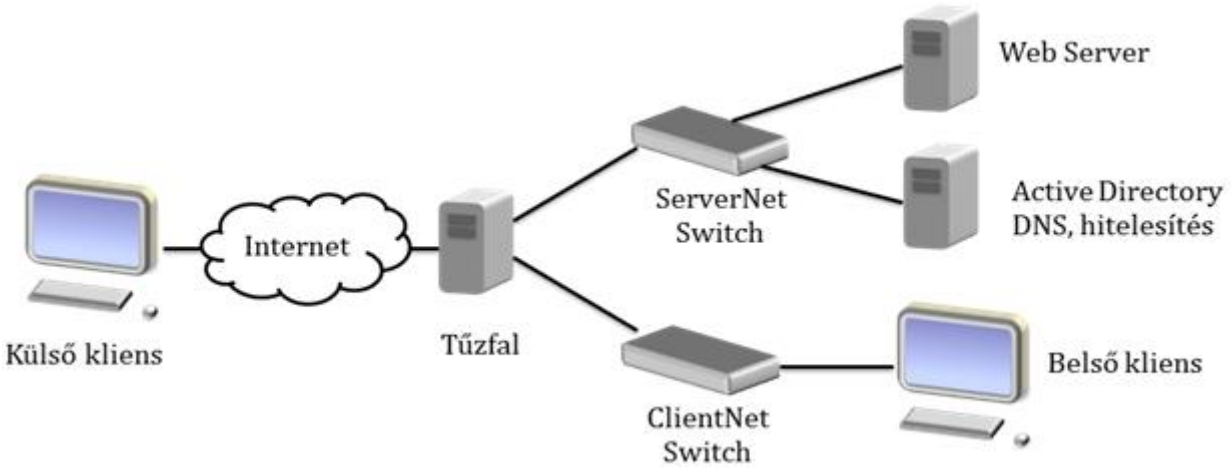
Hogy csökkentsük a kiesések lehetőségét, az infrastruktúránkat kiegészítjük a következő elemekkel. Veszünk öt darab netbookot, amik Linuxot futtatnak. Beállítunk egy linuxos szervert, amire egy openLDAP címtárat telepítünk. A linuxos szerverre rákötünk egy vezetéknélküli Access Pointot, a netbookok ehhez csatlakoznak. Az Access Point a felhasználók hitelesítését az openLDAP címtár segítségével oldja meg. Továbbá a szervereinket egy új switchen keresztül kapcsoljuk össze.

b. Rajzolja fel, hogy hogyan néz ki most az infrastruktúránk! (5 pont)
c. Adja meg, hogy hogyan néz ki egy adott netbookokra való sikertelen belépéshez tartozó hibafa. (5 pont)
d. Az új fejlesztéssel megjelent az a gond, hogy a felhasználók külön-külön léteznek az Active Directory és az openLDAP címtárban is. Javasoljon egy módszert a probléma megoldására! (2 pont)

3.2 Weboldal hibájának diagnosztizálása (2009. 06. 03.)

Adott a következő infrastruktúra részlet:

Az egyik gépen futó Web Server egy publikusan elérhető weboldalt szolgál ki, amelyen felhasználók be is léphetnek. A felhasználók hitelesítése egy másik gépen futó Active Directory szerverrel történik, ez a szerver egyben a DNS szolgáltatásért is felelős. A belső hálózati szegmensek valamint az internet kapcsolat között egy tűzfal található, ami csak a HTTP és DNS kéréseket engedi be, kimenő forgalomszűrést és címfordítást nem végez. A kliensek nem töltik be gyorsítótárból a weboldalt, a domain nevek feloldását viszont igen.



a) A rendszer üzemeltetője a következő hibajelentést kapja kívülről: egy távoli kliensen „nem jelenik meg a weboldal". Milyen okai lehetnek ennek a hibajelenségnek? Készítsen hibafát! (6 pont)

b) Az üzemeltető nekiáll ellenőrizni a bejelentett hibát és a belső kliensgépéről megnézi a weboldalt. A kezdőoldal megjelenik, ám amikor megpróbál belépni „500 Internal Server Error" hibajelzést kap. Milyen okai lehetnek ennek a hibajelenségnek? Készítsen erről is hibafát! (Itt nem kell jelölni negálásokkal azokat a komponenseket, amik leírt jelenség alapján biztosan jók, csak a lehetséges hiba- okokat) (4 pont)

c) Milyen közös okai lehetnek az a) és b) pontban leírt hibajelenségeknek? Mely komponensek hibája zárható ki? (6 pont)

d) Tegyen javaslatot arra, hogy mely komponenseket kellene monitorozni és (nagyvonalakban) hogyan, hogy a hibajelenség oka egyértelműen azonosítható legyen, illetve az üzemeltető előbb értesüljön az a) pontban leírt hibajelenségről, minthogy valamely felhasználó azt jelentené. (4 pont)

3.3 Video On Demand szolgáltatás (2009. 06. 17.)

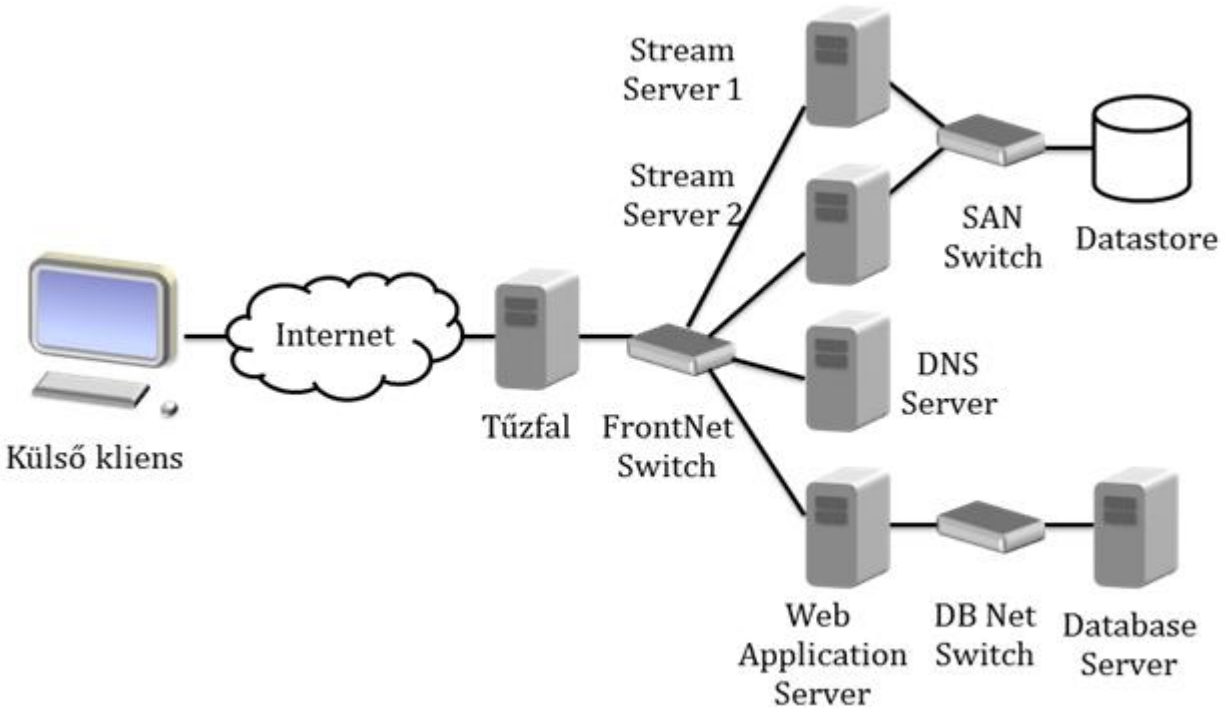
Egy VOD (Video On Demand) szolgáltató új üzemeltetőt vesz fel, akinek az első napján rögtön feladata is akad. Egy felhasználó azt a hibabejelentést teszi, hogy „néha azonnal működik a weboldalon a videó, néha viszont csak a kapcsolódásra vár". Viszont, többszöri próbálkozásra a videó megnyitása előbb-utóbb sikerül.

A rendszer felépítéséről a következő ábra áll rendelkezésére. Látható rajta, hogy a videótár szolgáltatás weblapját egy alkalmazásszerver szolgálja ki, ami egy adatbázist is használ. A videó folyamokat pedig a két Stream Server szolgálja ki. A videók egy SAN-on kapcsolt közös adattáron találhatóak.

a. Az új üzemeltető még nem ismeri a rendszert, csak annyit tud, hogy a két Stream Server valamilyen terheléselosztó fürtbe van kapcsolva. Milyen terheléselosztó megoldások lehetségesek ebben a környezetben? (4 pont)

b. Mi lehet az oka a bejelentett hibának? Készítsen hibafát arra az esetre, amikor a weboldal működik, de a lejátszás nem indul el! Mely komponensek hibája zárható ki, ha figyelembe vesszük, hogy a lejátszás néha elindul? (A hibafában most elemi hibaként kezelhető minden komponens hibája, nem kell külön OS, HW és alkalmazás hibát jelölni. Gyorsítótárazási mechanizmusokat az egyszerűség kedvéért nem kell feltételezni.) (8 pont)

c. Később bekövetkezik egy másik hiba is, ami következtében a video lejátszás teljesen leáll, de a weboldal továbbra is elérhető marad. Soroljon fel legalább 3 komponenst, aminek a hibája ezt okozhatja! Mindegyikhez írjon egy-egy technológiát, ami képes védekezni a hiba ellen! Rajzolja le a hibatűrési mechanizmusokkal kiegészített konfigurációt! (8 pont)



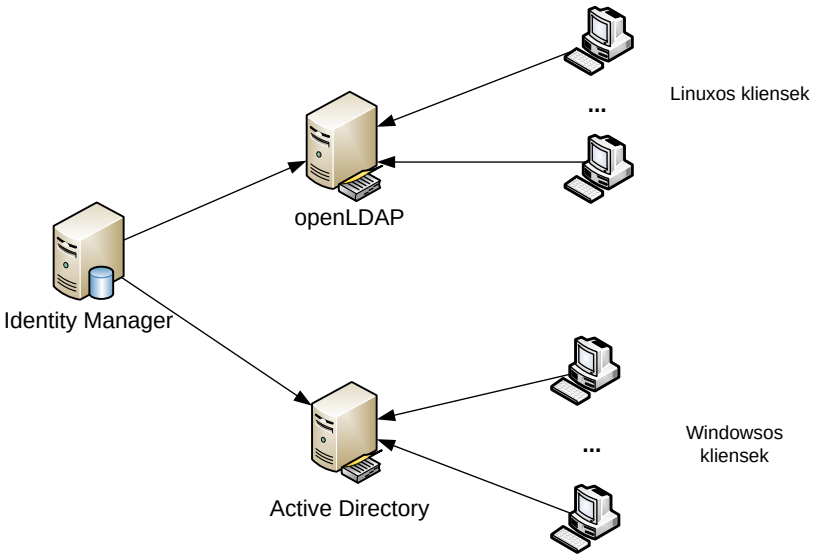
3.4 Identity manager használata (2010.06.14.)

Egy cégnél vegyesen linuxos és windowsos kliensek vannak. A kliensekre való bejelentkezést központi címtárak segítségével oldják meg, a windowsos gépek egy Active Directory tartományvezérlőhöz csatlakoznak, a linuxosok egy openLDAP szerverhez. Hogy ne legyen két különböző, független jelszava a felhasználóknak, a cég egy úgynevezett identity manager megoldást alkalmaz. Az identity managerben lehet a felhasználók adatait módosítani, és ez időnként frissíti ezeknek megfelelően a címtárakban lévő adatokat. A címtárakban közvetlenül ennek megfelelően nem lehet az adatokat módosítani. A rendszer logikai felépítését szemlélteti a lenti ábra. A számítógépek fizikailag egy közös nagy switchbe vannak bekötve.

a) Az FMEA módszernek megfelelően soroljon fel fontosabb meghibásodásokat és azok hatásait az egyes komponensekhez egy táblázatban (mindegyik komponenshez legalább kettőt adjon meg). (4 pont)

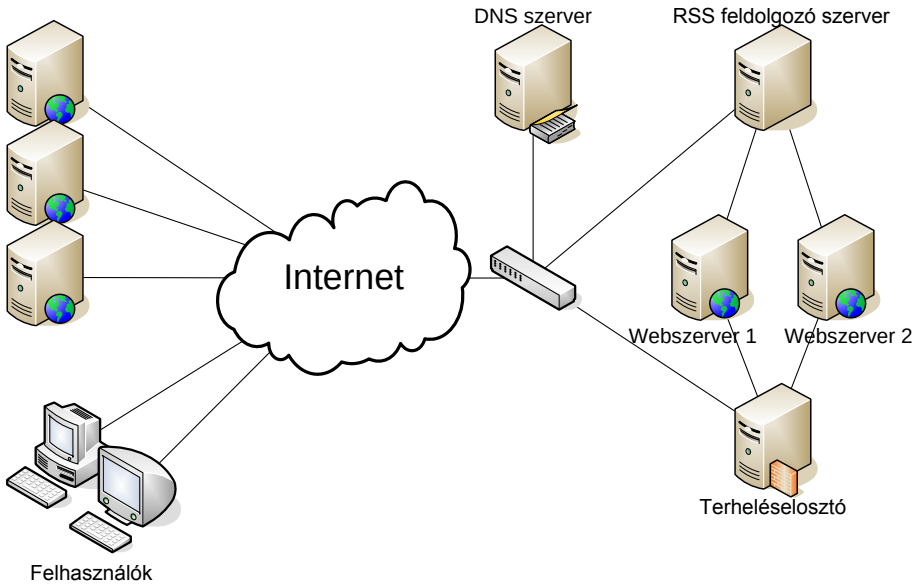
b) A fenti meghibásodásokat felhasználva készítsen egy hibafát arra az esetre, hogy ha a felhasználók azt a hibát jelentik be, hogy nem tudnak bejelentkezni a windowsos gépekre, pedig a linuxos gépeken minden működik rendesen. (4 pont)

c) A windowsos bejelentkezés elérhetőségével többször is probléma van. Milyen módszert javasolna a szolgáltatás rendelkezésre állásának növelésére? (2 pont)



3.5 RSS hírportál (2011.06.01.)

Vállalatunk egy olyan rendszer üzemeltetését végzi, mely a különböző hírforrásokon (RSS csatornákon) elérhető információkat összegyűjti, feldolgozza, majd kategorizálja, végül egy weboldalon elérhetővé teszi az így kialakított tartalmat a felhasználói számára a bérelt vonali internet-hozzáférésén keresztül.



Az RSS feldolgozó szerveren tároljuk egyszerű szöveges listában azon URL-eket, melyek által megadott hírcsatornákat fel kell dolgoznia a rendszernek. A szerveren egy időzített (Cron) folyamat 5 percenként végigmegy ezen a listán és egymás után sorban lekéri az aktuális RSS Feed-et a hírportáltól, majd feldolgozza, kategorizálja azt. Egy-egy RSS Feed letöltésének átlagideje a TCP (HTTP) kapcsolat felépítésének sebességét és az esetleges tengeren túli szerverek válaszidejét tekintve átlagosan 10 másodperc. Az összegyűjtött adatok alapján a szerver frissíti a két webszerveren helyileg futó, de egymástól teljesen független adatbázis tartalmát.

A felhasználók a kialakított tartalmat a terheléselosztó szerveren, mint szolgáltatáselérési ponton keresztül érik el. Ez a szerver egy egyszerű HTTP proxy, mely round-robin terhelésmegosztást valósít meg a két webszerver között. A terhelésmegosztás mellett ez a szerver még hibatűrő működést is biztosít, egy webszerver kiesésének érzékelése esetén a nem teljesített kérés(eke)t megismétli a másik

kiszolgáló irányába. Ő maga http szintű hibát csak akkor generál, ha egyik webszerver sem válaszol a kérésekre.

A feladat megoldása során a helyi hálózat hibáit elhanyagolhatja, valamint feltételezheti, hogy egyszerre nem válik elérhetetlenné az összes vizsgált hírportál sem.

a) Készítsen hibafát a következő eseményhez: a felhasználók böngészőjében nem megy az oldal, miközben feltételezhetjük, hogy a névfeloldás és a felhasználó oldali internet-hozzáféréssel nincs probléma. (3 pont)

b) Készítsen hibafát a következő eseményhez: a felhasználóknál megjelenő weboldal nem mutat friss tartalmat, a legújabb hír is már 15 perce jelent meg. (3 pont)

c) Milyen problémák merülnek fel abban az esetben a vázolt architektúrában, ha a vállalkozás növekedésével a kezdeti 100 felhasználó és 20 vizsgált hírportál száma tízszeresére emelkedik? Milyen adatokat, metrikákat vizsgálna és becsülhető értékük alapján milyen módosítási javaslatokat tenne? Soroljon fel legalább 3 metrikát, és tegyen javaslatot az architektúra javítására! (4 pont)

3.6 Ügyviteli rendszer (2011.06.08.)

Cégünk egy rendkívül népszerű ügyviteli rendszert fejleszt, mi pedig vezető terméktámogatási mérnökként a legproblémásabb esetek elhárításában segédkezünk. Hogy megdolgozzunk a nem kevés fizetésünkért, a következő probléma megoldásában kell segédkeznünk az egyik ügyfélnél. Az alkalmazásunknak van egy vastag kliense, egy skálázható, Java alkalmazásszerveren futó szerver oldali komponense és egy háttéradatbázisa. Az ügyfél bejelentése szerint a szegedi telephelyükön lévő kliensekről nézve van probléma (a pontos hibajelzés: „nem működik, és valami piros felkiáltójeles hibát dob fel néha”). Az ügyfél rendszergazdáját nem sikerült még elérni, korábbi kommunikációk alapján annyit tudunk, hogy az alkalmazásunkat a következő konfigurációban telepítették. A szegedi telephelyen csak kliensek futnak, ők egy VPN csatornán keresztül csatlakoznak a budapesti központ belső hálózatára. A központban jelenleg két alkalmazásszerver fut. Az adatbázisszerver hibái ellen pedig replikációval védekeznek (elsődleges és másodlagos példány fenntartása).

a) Az eddigiek alapján próbáljunk a rendszer felépítéséről egy ábrát rajzolni a főbb elemekkel és kapcsolataikkal. (2p)

b) Ábrázoljuk hibafa segítségével, hogy mik okozhatják a fenti hibát! (4p)

c) Milyen ellenőrzéseket végeznénk a rendszer különböző pontjain, hogy megpróbáljuk behatárolni, hogy mi a gond (milyen eszközzel, milyen sorrendben)? (3p)

d) A fenti ellenőrzések közül melyiket végeznénk el, ha közben megtudnánk, hogy a budapesti központban dolgozók elérik az ügyviteli rendszert? (1p)

3.7 MMO rendszer debütálása (2011.06.15.)

Egy napsütéses nyári napon végre eljön a nagy esemény ideje. Évek óta készültünk erre; az általunk fejlesztett MMO debütálására. Az esemény Facebook oldalára kb. tízezren jelentkeztek fel. A kliens alkalmazás már pár napja elérhető, de egyelőre csak pár százan töltötték le, szóval nagy roham várható. Előrelátóan a saját letöltő szervereinken kívül BitTorrenten is elérhetővé tettük az állományokat.

Az alkalmazás kliense TCP csatornán keresztül kapcsolódik a szerveroldalhoz. A szerver oldal belépési pontja a kliensek felől pedig egy központi terheléselosztó, amely fogadja a bejövő kapcsolatokat, és round-robin módszerrel kiválasztja azt a szervert, amely ki fogja szolgálni. Ez a kérés típusától függően letöltés esetén egy statikus webszerver, a játék működéséhez kapcsolódó kérés esetén egy alkalmazásszerver. Az alkalmazásszerverek mögött lévő adatbázist is fürtözve valósítottuk meg, a fürt igény esetén új tagokkal tovább bővíthető a teljesítmény és a rendelkezésreállás javítása érdekében.

a) Az eddigiek alapján próbáljunk a rendszer felépítéséről egy ábrát rajzolni a főbb elemekkel és kapcsolataikkal. (2p)

b) Az első pár óra jól telik, a kliensek letöltése zavartalanul folyik. Azonban, mikor a felhasználók elkezdenek tömegesen bejelentkezni, akkor az először csak sok ideig tart, aztán egy idő után már nem is sikerül. Ábrázoljuk hibafa segítségével, hogy mik okozhatják a fenti hibát! (4p)

c) Milyen ellenőrzéseket végeznénk a rendszer különböző pontjain, hogy megpróbáljuk behatárolni, hogy mi a gond (milyen eszközzel, milyen sorrendben)? (3p)

d) A letöltést kiszolgáló web szerver és a játékot kiszolgáló alkalmazásszerver szerepek többféle elrendezésben telepíthetők a rendelkezésre álló hardverekre. Ábrázoljon két elrendezést, ezek közül az egyiken azt, amely esetén a hálózattól függetlenül a letöltés akadályozhatja a játék kiszolgálását. (1p)

3.8 Nem működik a feladatbeadó rendszer! (2012.05.30.)

Nevenincs város egyetemének egyik tanszéke új, online feladatbeadó rendszert vezet be a hallgatói elégedettség és oktatói munka támogatásának növelése érdekében. A rendszer biztosítja a tanszék munkatársai számára a különböző feladatok kiírását, határidőinek beállítását, majd a beérkezett feladatok pontozását és értékelését. A hallgatók mindig meg tudják tekinteni a számukra kiadott feladatokat és azok elkészítése után a webes felületen lehetőségük van beadni azt. A rendszer megbízható működése a tanszék munkája szempontjából kritikus, mert a határidőket szigorúan veszik és rendszerleállás esetén ezeket nem lehet betartani sem hallgatói, sem pedig oktatói oldalról.

A hallgatói felületet a várható terhelés és rendelkezésreállási követelményeknek megfelelően egy terheléselosztó proxy mögött futó 3 webszerver szolgálja ki, miközben a metaadatok két redundáns adatbázisszerveren, míg a beadott fájlok pedig egy hálózati tárolón (NAS) kerülnek tárolásra. A proxy biztosítja, hogy bármely webszerver kiesése esetén a terhelés újra elosztódjon, és a felhasználói oldalon ne jelenjen meg semmilyen hibajelenség.

Biztonsági okokból az oktatói hozzáférés egy külön webszerveren keresztül történik, ami csak belső hálózatból vagy VPN-en keresztül érhető el (de ugyanahhoz az adatbázishoz és háttértárolóhoz kapcsolódik és onnan nyeri az adatokat, mint a hallgatói felület).

A két adatbázisszerver Master-Slave replikációt használ, azaz az adatokat minden esetben a Master szerver fogadja és a tranzakció zárása előtt a módosításokat a Slave rendszerre szinkronizálja. A Master kiesése esetén a Slave válik Master szerverré és átveszi a szolgáltatás hozzáférési pontot (IP-címet). A Slave kiesése esetén a Master egy ideiglenes tárolóba gyűjti a módosításokat, hogy amennyiben a Slave újra elérhető lesz, azokat át tudja adni. A NAS egy 3 lemezzel rendelkező RAID5 technológiával szerelt hálózati tároló, amit mindegyik webszerver elér és ki tud szolgálni.

Az egyik hallgató panaszkodott, hogy nem tudta beadni a feladatát. Azt már kiderítettük, hogy a határidőben még éppen benne volt, amikor próbálkozott, így most a feladatunk azt megállapítani, hogy mi okozhatta a meghibásodást.

a) Az eddigiek alapján próbáljunk a rendszer felépítéséről egy ábrát rajzolni a főbb elemekkel és kapcsolataikkal. (2 pont)

b) Ábrázoljuk hibafa segítségével, hogy mik okozhatják a fenti hibát! (4 pont)

c) Milyen ellenőrzéseket végeznénk a rendszer különböző pontjain, hogy megpróbáljuk behatárolni, hogy mi a gond (milyen eszközzel, milyen sorrendben)? (3 pont)

d) A fenti ellenőrzések közül melyiket végeznénk el, ha közben megtudnánk, hogy az egyik oktató kollégánk közben feltöltött eredményeket a rendszerbe, és ő nem tapasztalt leállást? (1 pont)

3.9 A pontos hiba: „Nem megy a levelezés” (2012.05.30.)

Vállalatunk minden munkája az alkalmazottak kommunikációjára épül, ezért egy nagy megbízhatóságú levelezőrendszert építettek ki. A rendszertervezésért felelős csoport vezetőjeként a mi feladatunk a komolyabb meghibásodások esetén történő hibadetektálás és az ebből fakadó javítási, módosítási javaslatok megtétele.

A felhasználói fiókok adatait egy adatbázisszerver tárolja, ami kizárólag a belső hálózaton érhető el. Innen veszi minden további szerver a postafiókok eléréséhez szükséges felhasználói neveket, jelszavakat és a levelek tárolására használható mappák elérési útvonalát. A levelek tárolása egy nagy megbízhatóságú tárolón történik, amit egy hálózati tároló egység (NAS) valósít meg. Ebben RAID5 technológiával 3 merevlemez foglal helyet. A levelek fogadását és a postafiókokban történő elhelyezését jelen pillanatban a két MX szerver (fogadó szerver) végzi, melyek teljesen egyformák és egyik kiesése esetén a másik ellátja feladatát.

A felhasználók a leveleiket asztali levelező klienssel IMAP protokoll segítségével, vagy a Webmail felületen keresztül érik el. Mind az IMAP, mind pedig a Webmail-t kiszolgáló szerver eléri a levelek tárolására használt tárolót és az adatbázis alapján hitelesíti a felhasználókat. A Webmail felület nem más, mint egy egyszerű webes levelező kliens, aki a fájlrendszerből felolvassa a leveleket és megjeleníti a böngészőben.

Az IMAP- és Webszerver funkciót szintén két-két, egyforma szerver valósítja meg, melyek egy-egy terheléselosztó proxy mögött foglalnak helyet. A proxy szerverek biztosítják, hogy bármelyik általuk menedzselt szerver kiesése esetén a forgalmat a másik felé terelje és a felhasználók oldalán ne jelentkezzen a hiba.

Egyik asztali klienst használó kollégánk pontos hibajelzése az, hogy „Nem megy a levelezés”.

a) Az eddigiek alapján próbáljunk a rendszer felépítéséről egy ábrát rajzolni a főbb elemekkel és kapcsolataikkal. (2 pont)

b) Ábrázoljuk hibafa segítségével, hogy mik okozhatják a fenti hibát! (4 pont)

c) Milyen ellenőrzéseket végeznénk a rendszer különböző pontjain, hogy megpróbáljuk behatárolni, hogy mi a gond (milyen eszközzel, milyen sorrendben)? (3 pont)

d) A fenti ellenőrzések közül melyiket végeznénk el, ha közben megtudnánk, hogy egy másik kolléga a Webmail felületen keresztül látja a legfrissebb leveleket is? (1 pont)

3.10 Adatfeldolgozás Hadoop fürttel és Amazon EC2-vel megfűszerezve (2012.06.06.)

(Megjegyzés: A vizsgán a feladat egyszerűbb verziója szerepelt, ott nem volt benne az EC2-es rész.)

Friss startup vállalkozásunk a következő megoldással jelent meg a piacon. Egy mobil alkalmazás segítségével a felhasználók közlekedési állapotinformációkat küldhetnek be (pl. dugóban állok, baleset van stb.), melyhez az alkalmazás csatolja a pillanatnyi sebességet és GPS-pozíciót. Cserébe a felhasználók megnézhetik az adatokból készült összefoglaló jelentéseket, és a rendszer segít nekik később a problémás útszakaszok elkerülésében.

Ehhez egy olyan háttérrendszert kellett kidolgozni, mely képes nagy mennyiségű adatot elosztott módon feldolgozni. A jelenlegi, kezdeti rendszerben a beküldött adatok egy adatbázis szerverbe kerülnek (a mobil alkalmazással és az adatok rögzítésének módjával most a feladatban nem kell foglalkozni). Az itt tárolt adatokat egy Hadoop fürt segítségével dolgozzuk fel. A Hadoop egy elosztott adatfeldolgozó megoldás[1]. A slave csomópontok futtatják az úgynevezett DataNode szerepet, ezzel egy HDFS nevű elosztott fájlrendszert valósítanak meg. Ezen kívül kellenek még master csomópontok, akik a NameNode szerepet (ez tárolja a fájlrendszer és a többi csomópont metaadatát), valamint a JobTracker szerepet (ez felelős a feladatok szétosztásáért) futtatják. Jelenleg egy master (amely

[1] A feladatban néhány helyen egyszerűsítettük a Hadoop valódi működését.

mindkét szerepet kiszolgálja) és három darab slave csomópontunk van. Az adatfeldolgozás a MapReduce elvet követi: a megkapott adatokat kis blokkokra bontják, ezeken a DataNode csomópontok külön-külön elvégzik a szükséges átalakításokat, számításokat, majd a végén összefésüljük az eredményeket. Minden blokkot több csomópont is tárol, és ha valamelyik DataNode kiesik, akkor az ahhoz rendelt befejezetlen feladatokat a JobTracker újra tudja osztani. Erre a keretrendszerre építettük rá a következő megoldást. Egy BIPortal (BI - Business Intelligence) nevű saját webes alkalmazás (ami egy külön gépen futó webszerveren van) indítja el este az új adatok feldolgozását. Ez áttölti a friss adatokat az adatbázisból a HDFS fájlrendszerre, majd létrehoz egy új feladatot a JobTracker-ben. Ezután a feladat elindul, a DataNode csomópontok feldolgozzák a beérkezett adatokat, majd az elkészült jelentéseket egy külön hálózati megosztásra másolják. Ezt a megosztást egy NFS szerver tárolja (mely az eddigiektől külön hardveren fut), a megosztásban minden naphoz egy-egy könyvtár tartozik. Ez a rendszer a kezdetekben jól működött, azonban mostanára vannak olyan napok, ahol túl sok adat érkezik be, és a rendszer nem képes másnap reggelre elkészülni az összes jelentéssel. Ez még csak néha fordult elő, így nem akarunk újabb hardvert venni, ehelyett Amazon EC2 virtuális gépeket használunk szükség esetén. Egy virtuális gépet futtatunk folyamatosan (CloudMaster néven hivatkozunk rá), amely igény esetén gyorsan tud újabb Amazon virtuális gépeket példányosítani, amiket felkészít, hogy Hadoop slave csomópontként üzemelhessenek. Ehhez a Chef konfigurációkezelőt használja, ez a virtuális gépek elindulása után elvégez minden szükséges beállítást. Mivel érzékeny adatokat tárolnak a DataNode-ok is, ezért a létrehozott virtuális gépek egy privát hálózatra csatlakoznak, és a CloudMaster felel azért, hogy egy VPN-csatornát építsen ki a saját infrastruktúránk belső hálózatának határán lévő gateway szerverrel, amin keresztül az EC2-es slave-ek a fürt többi részét elérik. Azt sikerült már kimérni, hogy a helyi infrastruktúránk kb. 1 GB adatot képes feldolgozni egy éjszaka alatt megbízhatóan, és minden egyes további fél GB-hoz egy újabb EC2-es slave csomópontra van szükség.

a) Rajzoljon egy ábrát a rendszer jelenlegi felépítéséről! (2 pont)

b) A belső hálózaton futó kliens gépről egyik nap reggel meg tudjuk nézni a korábbi jelentéseket, azonban az előző naphoz nem jött létre a megfelelő könyvtár, nincsenek arról jelentések. Ez a hibajelentés alapján mit tudunk a rendszer aktuális állapotáról, mi az ami valószínűleg működik? (1 pont)

c) Az a) feladatban felrajzolt rendszer architektúra alapján készítsen egy hibafát, ami strukturált formában ábrázolja, hogy a „nem készült el az előző napi jelentés” hibának mik lehetnek az okai! (4 pont)

d) A rendszer működését egyelőre szondázással vizsgáljuk, a következő egyszerű teszteket hajtjuk végre: 1) Hadoop master csomópont pingelése a belső hálózaton lévő kliensről, 2) BIPortal webhely kezdőlapjának lekérdezése, 3) NFS megosztáson lévő könyvtár tartalmának kilistázása, 4) CloudMastertől az aktuális futó EC2 példányok számának lekérdezése. Egy táblázatban adjuk meg, hogy melyik teszt melyik, a hibafában használt hibatípust képes detektálni (a táblázat sorai a tesztek legyenek, oszlopai a hibák)! (2 pont)

e) Milyen tesztekkel kéne még kiegészíteni a d) feladat készletét, hogy minden hibát tudjunk egyértelműen detektálni? (1 pont)

3.11 Webes alkalmazás Azure alapokon (2012.06.13.)

Egy kódolási versenyeket lebonyolító webes alkalmazás kifejlesztésén dolgozunk épp. A játékosok bejelentkezés után minden nap újabb feladatokat kapnak, azokra megoldásokat tölthetnek fel. A rendszer leellenőrzi a megoldások helyességét és erőforrás-igényét, majd ezek alapján pontozza a résztvevőket. Az eredmények alapján minden este újraszámolja a rendszer a különböző csoportok ranglistáját.

Haladva a korral rendszerünket a Microsoft PaaS számítási felhő megoldásának, az Azure-nak a segítségével valósítjuk meg. Az Azure-ban[2] az alkalmazások komponenseit szerepekbe (role) csoportosítjuk. Két típusú szerep létezik, a web szerep egy IIS-en futó webes kódot, a worker szerep pedig tetszőleges, általános kódot futtathat. Meghatározhatjuk, hogy az általunk definiált szerepekből hány példány (role instance) fusson. Az egyes komponensek sorokon (queue) keresztül kommunikálhatnak, ezek aszinkron üzenetküldést tesznek lehetővé két fél között. Az adatok tárolása SQL adatbázisban (SQL Azure, az SQL Server PaaS változata) vagy úgynevezett táblákban (Table, tulajdonképpen egy típusos kulcs - érték pár tároló) történhet.

A rendszerükben definiáltunk egy CodeContestWeb nevű web szerepet, ennek a példányai valósítják meg a webes felületet. A játékosok adatait és az aktuális ranglistákat egy CodeContestData nevű SQL Azure adatbázisban tároljuk. A nagyszámú beérkező megoldás adatait egy SubmissionsTable táblában tároljuk, melyet a felhasználó nevéből és a feladat azonosítójából képzett kulccsal tudunk indexelni. A megoldások értékelése időigényes feladat, ezért ezt egy külön, CheckerWorker nevű worker szerep példányai végzik. A CodeContectWeb példányai egy SubmissionQueue nevű soron keresztül értesítik a CheckerWorker példányait egy-egy új megoldás beérkezéséről, ilyenkor egy-egy új üzenetet raknak be a sorba. Ha elkészült egy megoldás ellenőrzése, akkor a worker példány beírja az eredményt a SubmissionsTable megfelelő elemébe. Az új eredmények alapján a ranglisták frissítését egy StatisticsWorker nevű szerep példányai végzik esténként, amelyek feldolgozzák a SubmissionsTable elemeit, majd módosítják a ranglisták adatait az adatbázisban.

a) Ábrázoljuk a rendszer logikai felépítését! (2 pont)

b) Az Azure hibatűréséről a következőket tudjuk. Egy adott szerep példányait az Azure mindig legalább kettő úgynevezett „fault domain”-ben futtatja (ezek az alatta lévő hardver olyan csoportjai, aminek nincsenek közös hibapontjai). Az SQL Azure és Table adatokat pedig legalább 3 példányba replikálja az Azure. Jelenleg minden szerepünkből két példányt futtatunk. Ezek az információk és az a) feladatban felrajzolt elrendezés alapján rajzoljunk egy hibafát, ami a „Nem frissült reggelre a ranglista” hibajelenség lehetséges okait ábrázolja. (4 pont)

c) Minden egyes a hibafában definiált elemi meghibásodáshoz adjunk meg egy módszert, hogy hogyan tudnánk annak a detektálását elvégezni! (2 pont)

d) Végezzünk egy gyors költségszámítást! Egy web vagy worker példány futtatásának ára $0,5 / óra. Ha a felhasználók nem tudják elérni a weboldalunkat, akkor az $1000 / nap bevételkiesést jelent számunkra. A hibafa analízisével és egyéb mérésekkel azt sikerült meghatároznunk, hogy ha két web példányt futtatunk akkor éves szinten 99%-os, míg ha három példányt futtatunk, akkor éves szinten 99,9%-os lesz a rendelkezésre állásunk (pontosabban a készenléti tényezőnk). Megéri-e a harmadik példányt futtatni? (A pontos eredmény kiszámolását nem várjuk el, elég csak a számolás mikéntjét megadni.) (2 pont)

3.12 Streaming szolgáltatás teljesítménytesztelése (2013.06.05.)

A streaming szolgáltatások egyre inkább átveszik a hagyományos kábeltévé szerepét, azonban ezek hazánkban nehézkesen érhetőek el. A piaci résen felbuzdulva saját streaming rendszer fejlesztésébe vágtunk kis cégünkkel. A piacvezető Netflix példáját követve mi is cloud környezetben fogjuk futtatni a rendszerünk legtöbb komponensét. A szolgáltatásnál kritikus a rendszer teljesítménye, így már a fejlesztés korai fázisában is megkezdjük a rendszeres teljesítménytesztelést. Ehhez a következő folyamatot dolgoztuk ki. A folytonos integráció (continous integration, CI) kiszolgáló minden éjfélkor letölti a forráskódkezelő rendszerből a kód aktuális változatát, majd elindít egy teljes fordítást. Sok külső komponenst használ a kód, és nem akarjuk a fordítási folyamat hosszát feleslegesen növelni ezek letöltésével, így beüzemeltünk egy belső tárhely (internal repository) kiszolgálót, mely napközben egyszer ellenőrzi a megadott külső függőségeket, és letölti a hiányzóakat vagy a frissebb verziókat. A CI szerver ehhez a belső tárhely kiszolgálóhoz fordul csak, az internetet nem éri el. Ha előállt a lefordított

[2] A feladatban az Azure működését némileg leegyszerűsítettük.

szoftver, akkor a CI kiszolgáló elindítja a teljesítménytesztet. Ehhez először a teszt környezetet egy privát cloud rendszerben hozza létre. A cloud vezérlő (cloud controller) API-ján keresztül létrehozza a szükséges virtuális gépeket. Ezek két csoportba oszthatók, az egyik csoport gépein a saját streaming alkalmazásunk szerver oldali komponensei futnak, a másik csoportban pedig a terhelést generáló kliens gépek. Az alap virtuális gép sablonok egy NAS tárhelyen vannak, ezeket másolja át a cloud vezérlő az egyes virtuális gépeket futtató hypervisorok lokális lemezére. Ha létrejöttek a virtuális gépek, a CI kiszolgáló belép rájuk, és rámásolja a szükséges komponenseket, elindítja a streaming rendszert, majd terhelést generál a kliensekkel, és közben folyamatosan gyűjti a rendszer különböző teljesítménymetrikáit. Egy-egy ilyen terheléses teszt általában 2 óráig tart, a befejeződése után a CI kiszolgáló elérhetővé teszi a webes felületén a tesztek eredményeit. Az összes kiszolgálónk jelenleg egy belső hálózaton található.

a) Gyűjtsük össze egy táblázatban, hogy melyik komponens milyen meghibásodása(i), okozhatják azt, hogy reggelre nem érhető el az előző éjszaka teljesítménytesztjének eredménye! (4 pont)

b) A felügyeleti rendszerünkben jelenleg a következő ellenőrzésekhez vannak riasztások felvéve: a CI kiszolgálóhoz intézett http kérés sikertelen, a cloud controller nem pingelhető, a NAS tárhely által kiajánlott kötetet nem lehet felcsatolni, az forráskódkezelő kiszolgálót nem lehet pingelni. Adjuk meg egy táblázatban, hogy ezek az ellenőrzések mely az a) pontban összegyűjtött meghibásodásokat képesek detektálni! (2 pont)

c) Specifikáljunk további ellenőrzéseket, amelyekkel az összes, az a) pontban azonosított meghibásodás detektálható, és ábrázoljuk ezek viszonyát a meghibásodásokhoz táblázatban! (3 pont)

d) Minimális-e az így előállt ellenőrzések halmaza? Miért? (1 pont)

3.13 Árvízvédelmi kommunikációs rendszer (2013.06.05.)

Nevenincs ország elhúzódó árvízi védekezésre számít, aminek fontos eleme a kor szellemének megfelelő kommunikációs rendszerek alkalmazása. Korábbi kellemetlen tapasztalatokból kiindulva a védekezés vezetői nem engedhetik meg, hogy a rendszer működésében fennakadások legyenek és emiatt anyagiakban vagy emberéletekben kár keletkezzen. A rendszer fontos eleme a publikus kommunikációs csatorna, amin keresztül a lakosságot informálják, valamint a belső, amin csak a települési vezetőknek és a gáton dolgozó területi felelősöknek juttatják el a védekezéshez szükséges információkat.

A rendszer várhatóan nagy forgalmat bonyolít le a kritikus időszakban, míg az év további részében csak a szükséges minimális forgalom fut majd rajta, így a vezetés az alapvető működéshez saját infrastruktúrát, míg a további skálázhatóság biztosítására felhő-alapú szolgáltatásokat vesz igénybe.

A publikus, webes elérés egy terheléselosztó proxy kiszolgálón keresztül érhető el, ami mögött alapesetben egy webszerver működik, de a kritikus időszakban a felhő-alapú rendszer szolgáltatójától bérelnek egy újabb proxy-t ami mögött 3 webszerver foglal helyet. A működés ekkor úgy zajlik, hogy az első proxy a terhelést arányosan az első webszerver (25%) vagy a felhő-alapú rendszer szolgáltatójának proxy-ja felé (75%) irányítja.

A rendszer által kiszolgált dinamikus adatokat redundáns adatbázisfürt tárolja, amit két adatbázisszerver biztosít Master-Master replikációs modellben, azaz bármelyik szerveren kezdeményezhetünk módosítást, ami a másik szerverre továbbítódik. A statikus adatok (fényképek, filmek) egy nagyméretű, elosztott hálózati tárhelyen foglalnak helyet, melyet 4 önálló, nagy tároló kapacitással rendelkező szerver biztosít teljesen elosztott rendszerben, azaz nincs kitüntetett Master szerver. Az összes webszerver ezen adatbázisfürthöz és hálózati tárolóhoz kapcsolódik, innen szolgálja ki az adatokat.

Biztonsági okokból a belső hozzáférés egy külön webszerveren keresztül történik, ami csak belső hálózatból vagy VPN-en keresztül érhető el (de ugyanahhoz az adatbázishoz és háttértárolóhoz kapcsolódik és onnan nyeri az adatokat).

Az egyik lakos panaszkodott, hogy nem tudta megnézni, mikor kerülnek lezárásra az alsó rakpartok a fővárosban. Azt már kiderítettük, hogy a helyi internetkapcsolata működött, így most a feladatunk azt megállapítani, hogy mi okozhatta a meghibásodást az állami rendszerben.

a) Az eddigiek alapján próbáljunk a rendszer felépítéséről egy ábrát rajzolni a főbb elemekkel és kapcsolataikkal. (2 pont)

b) Ábrázoljuk hibafa segítségével, hogy mik okozhatják a fenti hibajelenséget! (4 pont)

c) Milyen ellenőrzéseket végeznénk a rendszer különböző pontjain, hogy megpróbáljuk behatárolni, hogy mi a gond (milyen eszközzel, milyen sorrendben)? (3 pont)

d) A fenti ellenőrzések közül melyiket végeznénk el, ha közben megtudnánk, hogy az egyik területi vezető a mobiltelefonjáról időközben friss fényképeket töltött fel a rendszerbe, és ő nem tapasztalt leállást? (1 pont)

3.14 RSS-olvasó szolgáltatás (2013.06.12.)

Márciusban a Google bejelentette, hogy leállítja a Reader nevű RSS-olvasó szolgáltatását. A megüresedett piaci szegmensbe szeretnénk mi is betörni egy új rendszerrel. Az üzleti terv az, hogy ingyenesen tesszük elérhetővé minden felhasználó számára az alap alkalmazást, fizetni csak akkor kell, ha valaki extra modulokat vesz igénybe (pl. továbbosztás, Twitter becsatolása stb.). Ettől azt reméljük, hogy nagyon sok emberhez fog eljutni az alkalmazásunk – majd szépen sorban mindenki vásárol magának egy kisebb vagy nagyobb bővítést. Mielőtt viszont piacra dobnánk a szoftvert, szimulált kliensekkel teljes rendszerteszteket hajtunk végre rajta.

Az alkalmazás több különböző típusú klienst támogat, azok különböző protokolljaival. A szerver oldalt úgy terveztük és fejlesztettük, hogy egy-egy modulból többet is el tudunk indítani – akár különböző kiszolgálókon is. Jelenleg 2 extra modult implementáltunk, alapesetben mindegyik modulból 3 darabot indítunk, összesen 5 alkalmazás-szerveren egyenletesen elosztva. Ezen kívül mindegyik kiszolgálón fut egy-egy központi modul, ami közvetíti a forgalmat a megfelelő modulok felé és vezérli a terhelés elosztását, szükség esetén új modulok indításával és leállításával. Az alkalmazáshoz tartozó felhasználói adatok egy részét adatbázisban tároljuk, amit egy 3 gépből álló fürt szolgál ki, master-master replikációban. A nagyméretű adatokat (pl. képek, videók) egy 8 gépből álló elosztott fájlrendszeren tároljuk, ahol mindegyik hoszt egyenlő rangú, tehát akármelyik ponton el lehet érni a teljes fájlrendszert. A belső hálózathoz minden számítógép két interfésszel kapcsolódik, mind a kettő egy-egy nagy megbízhatóságú switch-hez csatlakozik. Kívülről DNS segítségével oldjuk meg a terheléselosztást úgy, hogy a domain-hez mind az öt alkalmazás-kiszolgáló IP-címe szerepel. (A belső hálózatunk és az internet között címfordítást és útvonalválasztást végző gépekkel nem kell most foglalkozni.)

a) Az eddigiek alapján próbáljunk a rendszer felépítéséről egy ábrát rajzolni a főbb elemekkel és kapcsolataikkal. (2p)

b) A szimulált terhelésnél egy adott kliensszám felett már egyre több kliens nem tud kapcsolódni a rendszerünkhöz. Tovább növelve a kliensszámot pedig végleg beadja a törülközőt – egyetlen klienst sem tud kiszolgálni. Ezek után már hiába csökkentjük a kliensek számát, már egyetlen egyet sem tudunk kiszolgálni. Ábrázoljuk hibafa segítségével, hogy mi történhet a háttérben! (4p)

c) Milyen ellenőrzéseket végeznénk a rendszer különböző pontjain, hogy megpróbáljuk behatárolni, hogy mi a gond (pontosan milyen eszközzel, milyen sorrendben)? (3p)

d) Feltéve, hogy a problémát nem az alkalmazásszerverek okozzák, javasoljon két változtatást az architektúrában, hogy legközelebb elkerülhessük a problémát! (1p)

3.15 Titkos társaság (2013.06.19.)

Évszázados hagyományokkal rendelkező titkos társaságunkat is elérte a modern kor, és a korábban a társaság kódexében vezetett tagsági hozzájárulásokat ezentúl elektronikusan szeretnénk tárolni.

Azonban a vezetőség eléggé konzervatív, így meg kell őket győzni arról, hogy attól, hogy nincs papírra leírva, még nem veszik el egy-egy tranzakció. Az IRF-ben tanultakat felhasználva a következő rendszertervet eszeltük ki. A tranzakciót feldolgozó komponensekből két különböző verziót fejlesztetünk majd le (az egyiket Java, a másikat Prolog nyelven implementáljuk). A két szoftverkomponenst külön-külön hardveren fogjuk futtatni. Minden beérkező kérést mindkét verzió külön-külön megkap majd. Erről egy szétosztó komponens gondoskodik. Az internetről egyedül ez a szétosztó érhető el, a többi gép egy belső hálózaton van rajta. A belső hálózaton két kapcsoló (switch) található, minden számítógép egy-egy interfésszel kapcsolódik az egyik és másik kapcsolóra, így a belső hálózati kapcsolatok is redundánsak. Hogy ne legyen a szétosztó egyszeres hibapont, egy aktív-passzív fürtöt rakunk alá. Miután a szétosztó elküldte a kérést mindkét feldolgozó komponensnek, azok egymástól függetlenül elvégzik a szükséges számításokat (mivel a társaság titkos, így itt komplex kriptográfiai algoritmusokat vetünk majd be). Az eredményeket elküldik egy, az eddigiektől független gépen futó szavazó komponensnek, ami összehasonlítja az eredményt, és ha megegyezik, akkor visszajelez az egyik feldolgozó komponensek, hogy beírhatja az eredményt az adatbázisba. Természetesen az adatbázis-kiszolgáló is egy külön hardveren fog futni. Ha az írás során hiba volt, akkor a feldolgozó komponens értesíti a klienst a tranzakció sikertelenségéről. Hogy biztos megmaradjanak az elmentett adatok, az adatbázis szerver 10 percenként elküldi a változásokat egy másik épületben lévő másodlagos példánynak (ehhez egy dedikált bérelt vonali kapcsolatot használ).

a) Rajzoljunk egy áttekintő ábrát, hogy hogyan nézne ki az elképzelt rendszer! (2 pont)

b) Hogyan kéne a szavazó komponensnek viselkednie? Adjuk meg egy táblázatban, hogyan kell döntenie a két feldolgozó komponens viselkedésétől függően! (2 pont)

c) Készítsünk egy hibafát, ami azt ábrázolja, hogy mikor fordulhat az elő, hogy a tranzakciót nem tároltuk le, de erről a klienst nem értesítettük! (4 pont)

d) Milyen egyszeres hibalehetőségeket tudunk azonosítani a hibafa segítésével a fenti hibajelenségre vonatkozóan? Mivel lehetne ezeket kiküszöbölni? (2 pont)