VÉDÉS SZÖVEG

1. Dia – Cím

*Köszöntés, téma

Az előadásomban először szeretném ismertetni azt a problémát, melynek egy megoldási lehetőségét az általam elkészített rendszer bemutatni hivatott, részletezem, hogy az alkalmazott technikák miért is jelenthetnek megoldási lehetőséget a problémára, kitérek az irodalomkutatási témámra, röviden összefoglalom, hogyan valósítottam meg a rendszert, majd pedig az elért eredményeket szeretném ismertetni.

2. Dia – Alapprobléma

Nagy általánosságban elmondható, hogy minden weben elérhető szolgáltatást valamilyen szerverrendszer üzemeltet, akár egy, akár több darab. Ezek a kiszolgáló rendszerek egyszerű esetben fixek olyan értelemben, hogy nem változtatják dinamikusan a hardvertulajdonságaikat vagy a szerverek számát. Ugyanakkor a terhelés, amit a felhasználók eredményeznek a rendszeren, mindig változik: vannak időszakok, amikor akár több nagyságrenddel többen használják, mint például éjjelente.

Ebből fakadóan, amikor kevesen használják a rendszert, az egy kérésre jutó fenttartási költség nagyobb lesz, ilyenkor kihasználatlan a rendszer, azaz feleslegesen sok szervert vagy feleslegesen erős hardvereket tartunk fenn a kiszolgálás céljából, ugyanakkor abban az esetben, amikor megnő a terhelés, az átlagos kiszolgálási idő nő, különösen rossz esetben pedig az egész rendszer összeomolhat. Növekvő terhelés esetén a kiszolgálás minősége romlik, esetleg a felhasználói élmény szintje esik vissza.

Az alapprobléma összefoglalva tehát az, hogy az egyszerű rendszerek nem tudják kezelni a változó terhelést.

3. Dia – Megoldás

Ezekre a problémákra a skálázódás jelenti ha nem is az egyetlen lehetséges, de mindenképpen magától értetődő megoldást. Az a probléma, hogy néha túl sok szerver fut, néha meg túl kevés? Hát változtassuk a számukat dinamikusan. Ez pofonegyszerűen hangzik, és valóban: nem nagy ördöngősség.

A skálázás azt jelenti, hogy valamilyen logika alapján változtatjuk a rendelkezésre álló erőforrások tulajdonságát. Azaz ha csökken a terhelés, akkor például szervert kapcsolunk ki, vagy csoportosítunk át más feladatok *kiszolgálására*, ha nő a terhelés, akkor pedig újabb és újabb szervereket indítunk el.

Nyilvánvalóan a döntés alapját a terhelés jelenti, tehát ha a mindenkori terhelés mértétkét meg tudjuk határozni, majd ennek lapján döntést hozni, egy hatalmas ugrással közelebb kerültünk egy hatékonyabb megoldáshoz annál, mint ha egy fix szerverünk lenne.

Az elosztott rendszer pedig arra kell, hogy függetlenül az éppen aktív kiszolgáló szerverek számától a rendszer mindig az elvárt működést tudja biztosítani. Azaz akár egy szerverünk van, akár száz, mindig helyesen kell működnie.

4. Dia – VPS

Irodalomkutatási témám a VPS-ek témaköre volt. Ezek olyan szolgáltatások, amelyek igénybevételével lehetőségünk nyílik szervereket bérelni, ahol elhelyezhetünk például olyan rendszereket, mint amilyet én valósítottam meg.

Ezen szolgáltatások esetében lehetőségünk van meghatározni, hogy pontosan milyen tulajdonságú szerverre fizetnénk elő: megadhatjuk, hogy hány maggal rendelkezzen a CPU vagy hogy mekkora legyen a memória.

A szakdolgozatomban kitérek ezen szolgáltatások hátrányaira is: bizonyos esetekben aggályos lehet adott kódot vagy adatot más cég szerverén, adott esetben más kontinensen elhelyezni, ám ezek a szolgáltatások jellemzően kifogástalan adatvédelmi és adatbiztonsági tulajdonságokkal bírnak.

Ugyanakkor nem elhanyagolható ezen szolgáltatások költségvonzata sem: a szolgáltatásnak is vannak költségei, de egy saját szerver üzemeltetése sincs ingyen, minden esetben mérlegelni kell a lehetőségek között.

Olyan szolgáltatókat kerestem, amik megfelelőek lennének a számomra: többek között fizikailag van közel szerverközpontjuk vagy éppen a díjszabás alapját az határozza meg, hogy milyen tulajdonságú szervert használ az előfizető és mennyi ideig.

5. Dia – Architektúra

A rendszer két főbb arcát is szeretném bemutatni az architektúra mellett ezen az ábrán:

- A rendszer egyik logikai megközelítése *a rendszer mint elosztott rendszer*. Ebben az esetben kiemelném, hogy a beérkező kéréseket egy kitüntetett szerver osztja szét az azok feldolgozására allokált szerverek között. Az, hogy hány ilyen feldolgozó/operatív szerver van, elosztott rendszer szempontjából lényegtelen. Az elosztott rendszer határát ez a kitüntetett master szerver jelenti, ezt a szervert érik el tulajdonképpen a felhasználók, akik ezáltal nem tudják explicit kiválasztani, hogy melyik operatív szervert használják éppen (nyilván mindegyik ugyanazt tudja).
- A rendszer másik megközelítése pedig a *rendszer mint skálázódó rendszer*. A master szerver figyeli a rendszer terheltségét, majd annak függvényében elindít egy újabb operatív szervert, leállít egyet vagy nem tesz semmit.

Nagyon szerencsésnek tartom magam azért, hogy a szakdolgozat témám 3 féléven keresztül ívelt át, mert az idő előrehaladtával szépen egységenként tudtam megvalósítani a rendszert. Első félévben témalaboron az irodalomkutatáson és az elméleten volt a hangsúly, önálló labor félévében létrehoztam egy elosztott rendszert, majd utolsó félévben alkalmassá tettem a rendszert az automatikus skálázódásra.

A rendszer tehát a szerverek számának dinamikus változtatásával reagál a változó terhelésre.

6. Dia – Szerverek

Az előbb elmondottak alapján tehát két szervertípust határoztam meg: master szerver és operatív szerver. Ezen típusok diszjunkt felelősségi körökkel bírnak. A master szervernek egyrészről üzemeltetnie kell az elosztott rendszer lelkét jelentő elosztott master adatbázist, illetve kiosztani a kéréseket, másrészről figyelnie kell a terhelést és döntenie a szerverek számáról.

A könnyű tesztelés érdekében úgy határoztam meg, hogy egy darab aktív kérésre egy darab operatív szervert biztosítok átlagosan. Egy percig átlagolja a terhelést, majd a periódus végén dönt a master szerver.

A működés során gyűjtött adatokat a master szerver egy státuszoldalon teszi elérhetővé.

Minden operatív szervernek ugyanaz a feladata: a beérkező kéréseket feldolgozása. Emelett az operatív szerverek feladata, hogy miután elindultak, beleintegrálódjanak a működő rendszerbe, jelezzék, hogy készen állnak a feldolgozásra.

Minden szerver ugyanolyan Ubuntu operációs rendszerrel rendelkezik, a rendszer által nyújtott szolgáltatást, a master szerver által publikált státuszoldalt és a kérések szétosztását egységesen az nginx webszerver üzemelteti, az elosztott adatbázist pedig mysql működteti.

7. Dia – Eredmények

A működő rendszer teljes mértékben megfelelt az elvárásoknak: a kék görbén látszik a terhelést reprezentáló kapcsolatok számának alakulása, a piros görbén pedig láthatjuk, hogy szépen követte a futó szerverek száma.

Érdemes megfigyelni az utolsó négy mért értéket: a terhelés drasztikus csökkenése időben elnyújtva jelenik meg a szerverek számának görbéjén. Minden döntés során legfeljebb eggyel változhat a szerverek száma biztonsági megfontolásokból.

8. Dia – Továbbfejlesztési lehetőségek

Jelen állapotban a rendszer rendkívül pazarló, nyilván abszurd az 1:1-es megfeleltetés a kapcsolatok száma és szerverek száma között. Jelenleg nem is a valós terhelést méri a rendszer, de mivel a rendszert modulárisan hoztam létre, könnyen kicserélhetők az egyes logikai elemek. Ez azt jelenti, hogy amennyiben más logika alapján szeretnénk mérni a terhelést, elég csak egy scriptet kicserélni, ami valóban méri a terhelést.

A demóalkalmazásom egy Üzenőfal alkalmazás, amely esetében a kérések feldolgozásának költsége infinitezimális, éppen ezért nem a processzorterhelést választottam a mérés alapjául. Ilyen képességű rendszerrel sokkal komplexebb szolgáltatásokat is lehetne üzemeltetni, ám ez egy Proof-of-Concept jellegű feladat volt, amellyel remélem sikerült bizonyítani, hogy ilyen problémákra létezik megoldás.

9. Dia – Bírálói kérdések

Még mielőtt az esetlegesen felmerülő kérdésekre válaszolnék, mindenképp szeretném megválaszolni a bírálói kérdéseket.