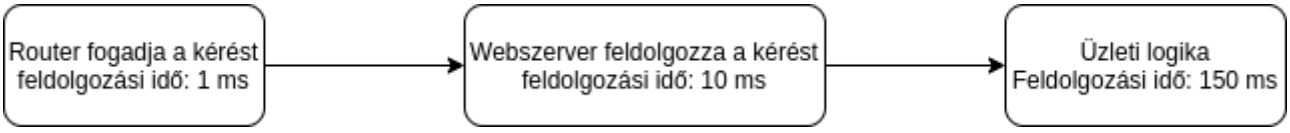
Szorgalmi feladat - Teljesítmény modellezés

Egy egyetem azon gondolkozik, hogy új tanulmányi rendszert vezet be. Az egyetemen folyó oktatás szervezése nagyon hasonlít a BME-hez, azaz a rendszert hallgatók és oktatók használnák. A hallgatók ezen a rendszeren keresztül vehetnék fel a tárgyaikat, vizsgáikat. Az oktatók felvihetnek tárgyakat, vizsgákat a rendszerbe, vizsgajegyeket írhatnak be a diákoknak.

Mivel nem akarnak azzal szembesülni, hogy a legelső tárgyfelvétel során teljesen összeomlik a rendszer így megkértek téged, hogy szimuláld a rendszert érő terhelést, és vizsgáld meg, ennek milyen hatása lesz a rendszerre.

A rendszer a webes kéréseket az alábbi módon szolgálja ki:



A kérést kezdetben a router fogadja, majd a webszerver feldolgozza, végül az üzleti logika valósítja meg a kért műveletet. Az üzleti logikáért az alkalmazásszerver felelős. Mind a három egység egyszerre egy kérést tud kiszolgálni és ha már egy kéréssel aktívan dolgozik akkor nem tud továbbiakat fogadni. Ha végez a kérés kiszolgálásával, de nem tudja továbbadni, mivel a következő egység még dolgozik, akkor várakozik, amíg az végez a saját kérésének a feldolgozásával. Természetesen várakozás közben sem tud új kéréseket fogadni.

Lehetőség van arra, hogy esetleg maximum egy kérés két művelet között várakozzon. De itt is bármely két egymást követő tevékenység között, maximum egy kérés várakozhat.

Feladat (5 pont)

Feladatod az lesz, hogy egy általad választott programozási nyelvben (C, C++, Java, C#, python) írj egy konzolos alkalmazást, amely leszimulál a fenti architektúrán egy 100 kéréses terheléses tesztet. Ez a száz kérés adott érkezési rátával egyenletesen érkezik. A tesztet több érkezési rátával is végezd el, majd X-Y diagrammon ábrázold az átbocsátást az érkezési ráta függvényében! (A diagramm elkészíthető excelben is, de más program, vagy programkönyvtár is használható hozzá)

Egy kis segítség

A mellékelt `main.c` egy példát mutat, hogy tudsz terhelést szimulálni. A példában egy egy-magos processzoron akarunk két számítást elvégezni egyszerre. Ezt úgy oldjuk meg, hogy a processzor mindig egy `P_switch` valószínűséggel taszkot vált, de ennek a taszkváltásnak van egy időköltsége. Ha a két taszk közül az egyikkel végzett, akkor többet nem vált taszkot, hanem egyszerűen befejezi a másik taszkot. A szimulációt 10000000-szer elvégezzük, majd kiírjuk, hogy mérésünk alapján a processzor átlagosan mennyi idő alatt végez a két számítással együttesen.

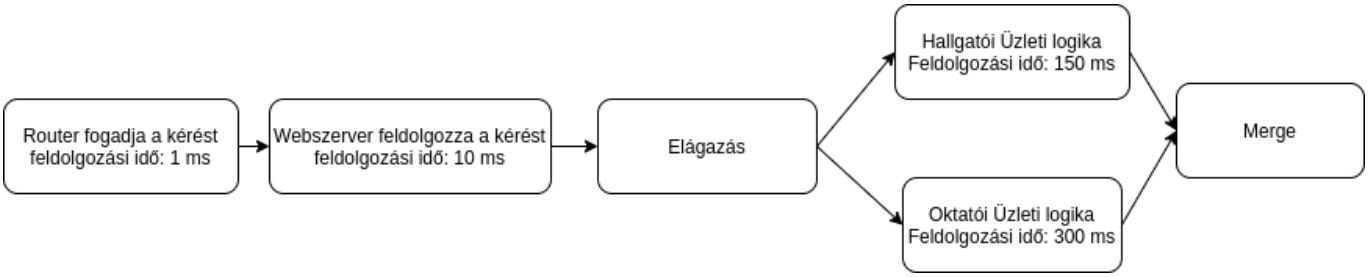
Bővítések

További pontokat az alábbi bővítések megvalósítására kapod. Az egyes bővítéseket egymástól függetlenül elvégezheted, kivéve ahol a feladat mást nem mond. Vizsgáld meg, hogy az egyes bővítések hogyan

módosítják a rendszer átbocsátását!

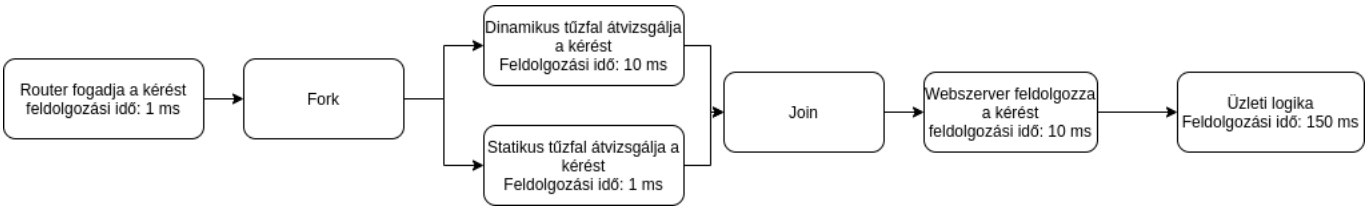
Maximális pontszámot akkor érnek a feladatok, ha azokat kellően általánosan oldottad meg (azaz a tanulmányi rendszertől eltérő architektúrák vizsgálatára is van lehetőség).

Elágazások (2 pont): A tanulmányi rendszerünket egyszerre használják hallgatók és oktatók is. A hallgatókat és az oktatókat más alkalmazás szerverek szolgálják ki. Egy bejövő kérés 0.1 valószínűséggel érkezik tanártól és 0.9 valószínűséggel diáktól. Készíts elágazás és merge elemeket!



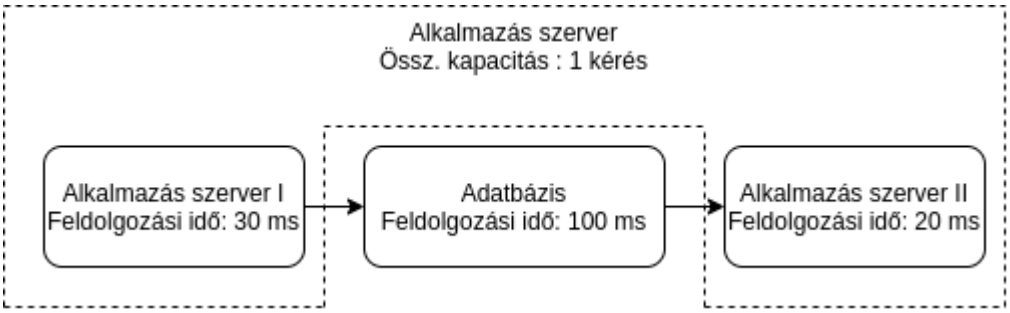
Kérések eldobása (2 pont, előfeltétel: elágazások): Ha sok hallgató rohamozza meg a rendszert, előbb utóbb a router és a webszerver arra fognak várakozni, hogy a hallgatói üzleti logika kiszolgálja a korábbi hallgatói kérést és addig nem tudnak új kérés elvégzésébe kezdeni. Így az oktatóknak is várniuk kell a kérés feldolgozására, pedig az ő alkalmazás szerverük kihasználatlanul várakozik. Erre megoldás az, ha felokosítjuk az elágazást, ami eldobja a kéréseket, ha a megfelelő alkalmazásszerver éppen nem tudja fogadni a kérést. Ha a kérés sikertelen, akkor a hallgató (vagy oktató) böngészője 5 ms múlva újra elküldi a kérést. A szimuláció addig tartson, amíg mind az összes hallgatói, mind az összes oktatói kérés sikerrel zárul. Vizsgáld meg ez egy általad választott magas érkezési ráta esetén mennyivel változtatja meg az oktatói kérések válaszidejének átlagát!

Párhuzamos végrehajtás (2 pont): A kéréseknek a nagyobb biztonság érdekében egy dinamikus és egy statikus tűzfalon is át kell menjenek, mielőtt őket a webszerver kiszolgálná. Ezeken tűzfalakon a kérés egyszerre mehet át. Valósíts meg fork és join elemeket!



Kapacitás (2+2 pont): Az alkalmazásszerverünk valójában az adatbázissal is kommunikál. Ha részletesebben megnézzük, akkor az üzleti logika tevékenységet tovább finomíthatjuk az ábrán látható módon. Az alkalmazás szerver elvégez pár műveletet, elküldi a kérést az adatbázisnak, majd végül elvégez rajta további műveleteket. Viszont az alkalmazásszerverben összesen továbbra is csak egy kérés tartózkodhat. A feladat 2 pontot ér eddig.

További két pontot akkor szerezhetsz, ha megvalósítottad korábban az elágazást, és mind a hallgatói üzleti logikát, mind az oktatói üzleti logikát külön-külön az ábrán látható módon három részre finomítod. A hallgatói üzleti logika feldolgozási idejeit az ábrán látod, az oktatói üzleti logika három lépése meg mind 100 ms ideig tartsanak. A két modul használja ugyanazt az adatbázist, azaz, ha az adatbázis már kiszolgál egy hallgatói kérést, akkor ne tudjon oktatói kérést fogadni, és fordítva.



Terheléselosztó elágazás(2 pont, előfeltétel: elágazás): A rendszer teljesítményét azzal szeretnénk növelni, hogy két webszervert alkalmazunk egy helyett. Egy okos terhelés elosztó mindig annak a webszervernek továbbítja a kérést, amelyik éppen szabad, de ha egynek sem tudja, akkor ő is várakozik. (Ha mindkettő szabad, akkor fix módon mondjuk az elsőnek továbbítja azt.)

Válaszidő ábrázolása(2 pont): Ne csak az átbocsátást, hanem az egyes kérések válaszidejét is tárold, majd rajzold ki box plot diagrammal, hogy az egyes érkezési ráták mellett hogyan változott a válaszidő!

Kódminőség (2 pont): Az alábbi két pontot arra kapod, ha jó minőségű, objektum orientált, könnyen bővíthető megoldást adsz be.

Beadás

Az elkészült kódot add be a hf portálon a Extra/iMSc feladat (Szimuláció, forráskód feltöltés) feladatra megoldásként, illetve töltsd ki a Extra/iMSc feladat (Szimuláció) nevű jegyzőkönyvet!

A feladat minden paramétere, ami ebben a leírásban nem szerepel, az a saját tervezői döntésedre van bízva. Viszont döntéseidet dokumentáld a jegyzőkönyvben!