

Féléves Feladat

Név: Molnár Attila

Neptun: ECJU2I

Téma:

Szaktervezéshez sok adathoz van szükségem, így a féléves beadandó során ezt a problémát szeretném megoldani, hogy ezeknek az adatoknak egy részét megszerezsem és valamilyen formában feldolgozzam. Részletezve a feladatomat, időjárási adatokat szeretnék letölteni, feldolgozni.

Az alábbi weboldalakon találtam segítségül API-kat ahol megtalálhatóak az elmúlt évek időjárási adatai.

- <https://developer.worldweatheronline.com/>
2 hónapig ingyenes elmúlt, elmúlt 10 év adatai, JSON-ban, napi 500 request és XML feldolgozás is lehetséges.
- <https://openweathermap.org/>
Ingyenes, viszont csak 5 évnyi adat tárolható, és óránként 60 query.

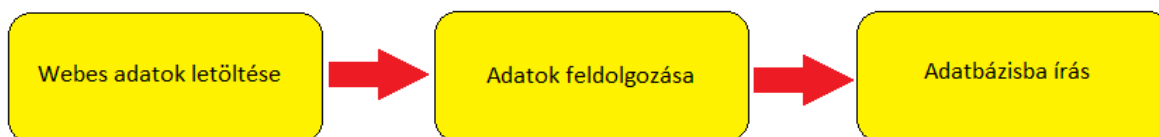
Természetesen, ha az idő engedi, mindkettő kipróbálható. Mindkettőből szükségünk van a következő adatokra: Hőmérséklet (itt lehet a maximum, a minimum, de akár egy nap átlag hőmérsékleteket is megtudhatjuk, ha szeretnénk), szélereősség, csapadékmennyisége, mindegyik valamilyen szám típusként lesz eltárolva.

Rendelkezésre álló architektúra: Intel Core i5-5200U Processzor, # of Cores 2 # of Threads 4.

Lehetséges megoldások

Nulladik megoldás:

Feladat szerint, előbb az adatokat betölteni és mikor ez elkezdődik megkezdjük azok feldolgozását.




Ez tök jó de ettől ez még nem lesz párhuzamos, szekvenciális marad a végrehajtás, viszont ezeknél a vázlatos részeknél lehetséges a párhuzamosítás.

Első megoldás:

A letöltést és az adatbázisba írást egy szál kezeli, olvasásra meg mindig másik szálat indítunk, így több felhasználó tudja olvasni az aktuális adatbázist. Ez jól működhet akkor, ha egyből meg is jelenítjük és egyszerre több felhasználó olvashatná ezeket az időjárási adatokat.

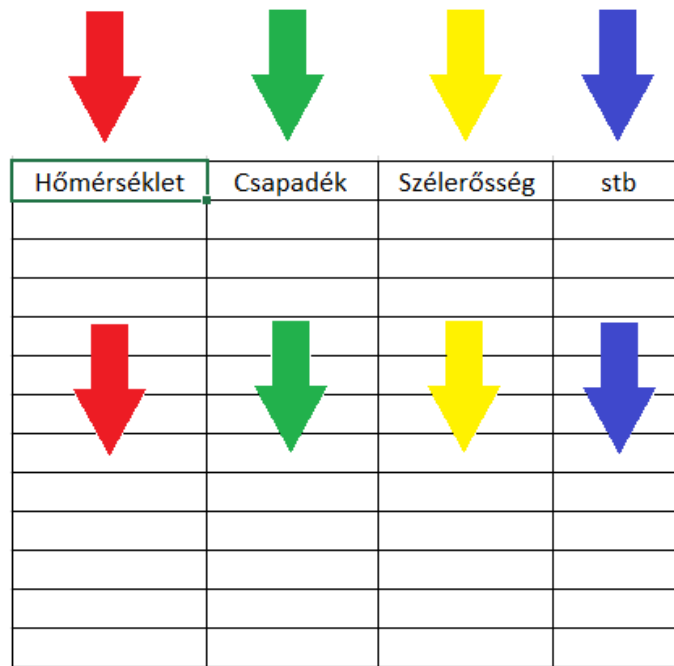
Második megoldás

Különböző adat típusokat, értsd hőmérséklet, szél erősség, csapadékmennyiség, részenként feldolgozni, azaz.



Hőmérséklet	Csapadék	Szél erősség	stb

A Rugalmasságot nézve így eléggé durván szemcsézett lenne a feladat. Finomítható a szemcsézettség, csak az adatokra, esetleg úgy, hogy az egyes sűrítjük a mintavételezést, de akkor az overhead nőne meg túlságosan.



Kódban is nagyobb káoszt okozhat, rontva annak olvashatóságát, karbantarthatóságát.

Évenként kérek egy-egy lekérdezést. Intervallumként állítom be, hogy például 2011.01.01-től 2011.12.31.-ig kérje minden egyes napnak a kívánt időjárási értékeit.

Ha 10 év adatait szeretnék megszerezni az 4 szárra lebontva azt jelentené, hogy 2 szár valamivel többet dolgozik, így van némi overhead, azonban elegánsabb évenkénti lehívás, ugyanis így egyből évenként külön táblákba el is menthetők ezek az adatok. Elsőre számomra ez a megoldás tűnik a leghatékonyabbnak. Nincs túl sok részre felbontva, logikusan évenként kezdem el kezelni az adathalmazt és végig aszerint is dolgozom fel.

Megvalósítás

A végső megoldásomban végül a worldonlineweather.com weboldalt használtam. Itt volt lehetséges a dátum szerinti kéréseket küldeni, valamint a legtöbb adat is innen volt elérhető. Egyetlen problémája, hogy egy kérésen belül egyszerre csak egy hónap adatai szerezhető meg. Három megoldásomat hasonlítottam össze ebből egy szekvenciális és kettő párhuzamos megoldás. A célom az volt, hogy a szekvenciális megoldás megvalósítása és néhány mérés után felgyorsítsam a program működését.

Elsőként a szekvenciális megoldást valósítottam meg, majd megvizsgáltam, hogy lehet-e párhuzamosítani. Ezután a API sajátosságait figyelembe véve, hasonló megoldást választottam párhuzamosításra, mint a tervezés során felvezetett harmadik megoldásom, ahol évenként dolgozom fel a kívánt adatokat. Mivel minden egyes hónapra külön kérést kell küldeni, ez a rész felgyorsítható ha például egy 10 éves időintervallumban az év kezdésektől kezdünk feldolgozni és nem az egész intervallum elejétől csak.

Szekvenciális megoldás

A soros megoldásomban csupán a későbbi összehasonlítás érdekében arra voltam kíváncsi, hogy mennyi időbe telik feldolgozni az adatokat. Első körben csak egy listába behelyezve mentettem el minden adatot, és mindenféle párhuzamosítás és szinkronizáció nélkül futtattam a programom. A beállításokban 2010.01.01 – 2016.-12.31 intervallumot adtam meg, ez 84 kérést jelentett és itthoni mérések során nagyjából 40 másodperc alatt végzett így a letöltésekkel.

Tesztelésnél egy InputGenerator osztálytól kapja a megfelelő éveket, ahol a „startdates” string tömbben érhető el a hónapok kezdtem, ennek megfelelője egy „endates” string tömb, ahol a hónap utolsó napja található, ezek a requestekhez szükségesek. Ez még a kezdeti lépéseknél lett létrehozva, a párhuzamos megoldásnál ennél valamivel jobb megoldásként Dictionaryben vannak páronként.

Szemafor párhuzamos megoldás

A szemafor megoldás során minden évenként indítottam egy-egy feldolgozási folyamatot. Érdekes 8 éve időjárási adatait elkezdeni, így elméletileg kiegyenlített lesz a műveletvégzők kihasználtsága. A feldolgozás (csak a letöltés) 30 másodpercen belül végzett.

Master – Worker megoldás

A Master feladata összeszedni a szükséges évek időjárási adatai. Ez a programban egyszerűen megadom a megfelelő gyűjteményt, és abban meg van, hogy mettől meddig szeretnék időjárási adatokat megkapni. A Worker feladata ezeknek az éveknek adatainak a letöltése, valójában most csak ők dolgoznak mivel nincs szükség az évek begyűjtésére.

A feldolgozás itt is csak a letöltésre érvényes, ez esetben is hasonló megoldást kaptam, mint a szemafor esetén tehát 30 másodpercen belül sikerült mindez. Legrosszabb eredményeket vettem figyelembe, ennél voltak lényegesen gyorsabb megoldások is. Néhány esetben 20 másodpercen belül is sikerült végrehajtani ugyanazt a műveletet. Összességében elmondható, hogy sikerült felgyorsítania a programot.