Eötvös Loránd Tudományegyetem

Informatikai Kar

Programozási Nyelvek és Fordítóprogramok Tanszék

Interpoláció osztott rendszereken

Tejfel Máté
egyetemi tanár

Cselyuszka Alexandra
Informatika Bsc

Budapest, 2015

Tartalomjegyzék

1. fejezet

Bevezetés

"A gyakorlatban soksor felmerül olyan probléma, hogy egy nagyon költségesen kiszámítható függvénnyel kellene egy megadott intervallumon dolgoznunk. Ekkor például azt tehetjük, hogy néhány pontban kiszámítjuk a függvény értékét, majd keresünk olyan egyszerűbben számítható függvényt, amelyik illeszkedik az adott pontokra." [1]

A szakdolgozatom célja ezekre a problémákra megoldást adni elosztott környezetben.

1.1. Feladat elemzése

Adott ponthalmazokból kívánunk egy közelítő polinómot becsülni. Ezeket különböző interpolációs technikával meg tudjuk adni, ki tudjuk számolni. Több interpolációs technika létezik, melyekből könnyen meg tudunk adni akár több polinómot is egy adott ponthalmazhoz.
Ezekkel a számításokkal előfordulhat, hogy lassan futnak, főleg ha több inteprolációt kívánunk egyszerre számolni. Ebben az esetben optimálisabb több gépen számolni a különböző ponthalmazokat.

Ebben a feladatban egy speciális megvalósítása lesz ennek a számításnak.
A grafikus része egy weboldal, melyen szerkeszthetjük a ponthalmazokat. A számítás részét egy szerver végzi amely figyeli a felcsatlakozó gépeket. Amikor kap egy számítandó adathalmazt, akkor több gép segítségével kiszámítja az eredményt. Ha minden részfeladat végzett, akkor vissza küldi a weboldalra, ahol az eredmények megtekinthetőek grafikus formában.

1.2. Feladat megvalósítása

A **grafikus felület** egy weboldal, mely javascript-ben és HTML-ben van megvalósítva. A felületen egy listát tekinthetünk meg, ahova több ponthalmazt is felvehetünk. Mentés hatásásra az értékek a háttérben eltárolódnak. A ponthalmazok közül választhatunk egyet, amely betölődik felületre.
A szerkesztő felület egy táblázatból és egy grafikonból áll, emellett még a különböző speciális számításra vonatkozó tulajdonságok (interpoláció típusa) valamint a grafikonon való megjelenítéshez tartozó tulajdonságok (polinóm pontosság, megtekintendő intervallum) is szerkeszthetőek.
Ha befejeztük a halmazok szerkesztését elküldhetjük a számítani kívánt értékeket a szerver felé.

A **szerver** feladata hogy figyelje a felületről érkező adatokat. Ha az adathalmaz megérkezett, akkor a szerver kibontja az adatokat egy JSON-ból, és elindítja az elosztást.
Az elosztáshoz a szerveren el kell indítani egy figyelő folyamatot amelyre lehetősége van egy külső gépnek felcsatlakozni. Amikor a szerveren indul egy számolás a felcsatlakozott gépeket lekérdezi, majd a feladatokat szétosztja.
A szerver megvalósítása és a gépekre való szétosztás Erlang-ban lett megvalósítva. A JSON feldolgozásához Mochi-JSON lett alkalmazva. A feldolgozás után az adathalmazon végig megyünk és azok alapján felparaméterezzük, és meghívjuk a számítást végző függvényt.
A számításhoz használt maximális gépek száma paraméterként megadható, de a tényleges számítást csak annyi gépen tujduk maximálisan végezni ahány gép felcsatlakozott a számításhoz.

A **számítás** megvalósítása C++ nyelven történt. A paraméterek alapján a Lagrange-féle, Newtone-féle, Hermite-féle interpolációs technikák közül eldönti melyik esetet használja.
Valamint inverz interpolációt is választhatnak a Lagrange vagy a Newton interpoláció esetén.
A programban implementálásra került egy egyszerű polinóm szorzás és össze adás, valamint az inteprolációkhoz szükséges függvények. Lagrange számítás a polinóm műveletek és a képlet felhasználásával valósult meg. Newton és Hermite esetén a kapott adatokból először a kezdő mátrixot kell legenerálni, majd kiszámítani.
Abban az esetben ha Newton vagy Lagrange polinómot számolunk nem vesszük figyelembe a derivált pontokat, viszont figyelmbe vesszük ha inverz számítást kívánunk végezni.

2. fejezet

Felhasználói dokumentáció

2.1. Bevezetés

2.2. Telepítési utmutató

2.2.1. Rendszer követelmények

2.2.2. Segédprogramok telepítése

2.2.3. Szerver és segédgépek üzembe helyezése

2.2.4. Használati útmutató

Weboldal

Szerver

3. fejezet

Fejlesztői dokumentáció

3.1. Megoldási terv

A program 3 fő részből áll a Weboldalból, az Elosztott rendszerből és a Kalkulátorból.

3.2. Weboldal

Weboldal felépítése HTML és JavaScript segítségével valósult meg. Egy oldalból áll melyen a felhasználó össze állítja a neki szükséges adathalmazt. Új adathalmazokat hozhat létre, a régieket szerkesztheti. A háttérben JSON-be formálódnak az adatok, melyeket a felhasználó is láthat, ha debug-módban lép be. Ha a felhasználó végzett egy gombra nyomással a program legenerálja a szükséges JSON-t.

3.2.1. Felépítés

3.2.2. Fontosabb objektumok

3.2.3. Kommunikáció

3.2.4. Tesztelési terv

3.3. Elosztott rendszer

Elosztott rendszer Erlang-ban lett megvalósítva. Az elosztást interpolációnként végezzük, vagyis annyi node-ot hozunk létre amennyi interpolációt kívánunk egyszerre kiszámítani.

3.3.1. Web-szerver kommunikáció

Adat feldolgozás Az elosztott rendszer először kap egy JSON adathalmazt melyből kinyeri a neki szükséges adatokat, és átkonvertálja.

3.3.2. Gép-szerver kommunikáció

3.3.3. Elosztás folyamata

3.3.4. Tesztelési terv

3.4. Kalkulátor

A Kalkulátor részben számítódik ki egy-egy interpolációnak az ereménye. A megkapott adatok alapján számol, ha kell létre hozza a kezdő mátrixot, kiszámolja az eredmény mátrixot, majd annak segítségével kiszámolja a polinómot.

3.4.1. Fontosabb számítási függvények

DArray interpolateMain

Kívülről meghívandó fő függvény mely elosztja és konvertálja a részeket

DArray &x : Az x pontok listája

DMatrix &Y : Az x pontokhoz tartozó y pontok halmaza

string type : Interpóláció típusa: lagrange, newton, hermite

bool inverse : Inverz interpoláció kell-e

3.4.2. Elosztott rendszerrel való kommunikáció

Az elosztott rendszerben hívódó számítást Erlang - erl_nif"-el sikerült megoldanom. Az ezzel kapcsolatos dolgokat az Calculator/erlang.cpp tartalmazza.

3.4.3. Tesztelési terv

A tesztelést folyamatosan végeztem a minta adatok alapján. A függvények implementálása közben ezekre a minta adatokra meghívtam, majd ezekkel számoltam. A teszteléshez a logTest.cpp fájlban található függvényeket alkalmaztam.

Irodalomjegyzék

[1] Gergó Lajos: Numerikus Módszerek, ELTE EÖTVÖS KIADÓ, 2010, [329], ISBN 978 963 312 034 7

[2] http://www.erlang.org/doc/man/erl_nif.html 2015

[3] https://www.sharelatex.com/learn/Sections_and_chapters 2015

[4] https://github.com/mochi/mochiweb/blob/master/src/mochijson.erl 2015

[5] http://tex.stackexchange.com/questions/137055/lstlisting-syntax-highlighting-for-c-like-in-editor 2015