Osztott rendszerek specifikációja és implementációja
1. Beadandó feladat dokumentáció

Iván Gergő
IVGRAAI.ELTE
ivgraai@gmail.com

2011. december 01.

1. Feladat

[asszociatív függvény kiszámítása] Szeretnénk meghatározni n mátrix szorzatának minden részeredményét. A mátrixokat fájlból töltjük fel, a szorzatokat a szabványos kimenetre írjuk ki. A programot C++ nyelven implementáljuk a pvm könyvtár felhasználásával.

2. A feladat megoldása absztrakt szinten

A feladatot három részre bontjuk: a mátrixok beolvasására, a szorzat kiszámítására és kiírására. A három részre adott megoldást szekvencia programkonstrukcióval kapcsoljuk össze. A második rész megoldását vissza akarjuk vezetni, egy az asszociatív függvény kiszámításának tételére. Ehhez az szükséges, hogy az általunk használt függvény, a „szorzás", asszociatív legyen. Az első és a harmadik rész megoldása elemi művelet.

2.1 A második részfeladat absztrakt megfogalmazása

Rögzítsük az n, $k \in \mathbb{N}_0$ számokat, és tekintsük a V = vector([1..n], vector([1..k], $\mathbb{Z}$)) típust.
Az állapot- és paramétertér legyen: $A = V \times V \times V$ $\quad B = V \times V$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad v_1 \quad v_2 \quad v_3 \qquad\quad v_1' \; v_2'$

A paramétertér pontjaihoz a feladat kételemű halmazokat rendeljen, amit a h_1 és h_2 futóindexszel jelölünk: F(b) = {h_1, h_2}. Az alábbi kikötéseket tesszük egy tetszőleges h_1 és h_2-re. Legyen Q = ($v_1 = v_1' \wedge v_2 = v_2' \wedge$

$v_1'.dom_2 = v_2'.dom_1 \wedge v_3.dom_1 = v_1'.dom_1 \wedge v_3.dom_2 = v_2'.dom_2$).

- $Q \in INIT_h$
- $Q \in TERM_h$
- $(\forall i \in [1..v_3.dom_1]:(\forall j \in [1..v_3.dom_2]:$
  $(\forall k \in [1..v_1'.dom_2]:(v_{3i,j} = v_1'_{i,k} * v_2'_{k,j})))) \in FP_h$

2.2 A visszavezetéshez használt tétel

Legyen H egy tetszőleges halmaz, amelyen adott egy $\circ : H \times H \to H$ asszociatív művelet. Definiáljuk az $f : H^* \to H$ függvényt, a művelet sorozatokra való kiterjesztését a következő módon:

$$f(\langle a_1 \rangle) = a_1$$
$$f(\langle a_1, a_2 \rangle) = a_1 \circ a_2$$
$$f(\langle a_1, a_2, \ldots, a_n \rangle) = a_1 \circ f(\langle a_2, \ldots, a_n \rangle)$$

Adott $a \in H^*$ mellett a $G_a : [1..|a|] \to H$ függvényt definiáljuk a következőképpen: $G_a(i) = f(\langle a_1, \ldots, a_i \rangle)$. A feladat a G_a függvény helyettesítési értékeinek kiszámítása.

Az a sorozatot és a G_a függvény helyettesítési értékeit reprezentálhatjuk egy-egy vektorral. Így a következő formális specifikációt írhatjuk fel:

$A = G \times G$, ahol $G = vector([1..n], H)$, ha $n = |a| \geq 1$
$a \quad g$
$B = G$
a'

- $Q = (a = a') \in INIT_{a'}$
- $Q \in TERM_{a'}$
- $(Q \wedge \forall i \in [1..n]:(g(i) = f((\langle a_1, \ldots, a_i \rangle))) \in FP_{a'}$

2.3 A feladatunkat megoldó absztrakt program

2.3.1 Lemma

A függvények kompozíciója asszociatív.

2.3.2 Bizonyítás

Két függvény akkor egyenlő, ha minden helyen megegyeznek.

$$\big(f \circ (g \circ h)\big)(x) = f\big((g \circ h)(x)\big) = f\Big(g\big(h(x)\big)\Big).$$

$$\Big(\big((f \circ g) \circ h\big)\Big)(x) = (f \circ g)\big(h(x)\big) = f\Big(g\big(h(x)\big)\Big).$$

A megfeleltetés a feladatunk és a tétel között a következő: $\mathbb{Z} \times \mathbb{Z} \sim H$, *mátrix szorzás* $\sim \circ$.

Igazolnunk kell még, hogy a mátrixokon értelmezett *szorzás* művelet asszociatív, azaz `*(M, *(N, K)) = *(*(M, N), K)`, feltéve hogy az összes szükséges szorzást el lehet végezni. Azaz, ha $M \in \mathbb{Z}^{n \times m}$, $N \in \mathbb{Z}^{m \times k}$ és $K \in \mathbb{Z}^{k \times l}$. Ez következik `2.3.1 Lemma`-ból.

3. Leképezés a PVM architektúrára

Az absztrakt program minden utasításának megfeleltetünk egy taszkot a konkrét programban, de még ezen kívül is lehetnek további taszkok. A `g` vektor elemeit elosztjuk az egyes taszkok között – minden taszkhoz a sorszámának megfelelő vektor elemet – egy mátrixot – rendeljük. Futás közben az egyes taszkok üzenetküldésekkel tudják egymással megosztani ezeket az értékeket és a részeredményeket, de mindig csak azokat, amikre az adott taszkoknak szüksége van a számításának elvégzéséhez. Ez lépésenként pontosan egy érték lesz.

- A taszkok létrehozása így történik. A legelsőt a felhasználó hozza létre, azzal, hogy elindítja a programot. Ekkor paraméterként átad neki egy fájlnevet, ami tartalmazza a mátrixokat megadott formában[1]. A mátrixok beolvasása után a program létrehoz annyi taszkot, ahány darab mátrix van a fájlban. Ezek után elvégzi a vektor elemeinek szétosztását is. A taszkok lefutása után, a szülő megkapja a taszkok eredményeit.
- Miután a taszkok létrehozása megtörtént, megkezdődhet az absztrakt programnak megfelelő kód végrehajtása. Egy taszk addig él, amíg vagy van még valakinek küldenie az ő eredményét, vagy van még kitől fogadni egy mátrixot. Ha van még kitől kapnia, akkor számol egy új értéket, vagyis végrehajtja a szorzást a régi mátrix változó értékébe.

Az implementációban nem feltételeztük, hogy az asszociatív művelet kommutatív is, sőt mátrixok szorzása – mint ismert – nem az. A taszkok többsége megfeleltethető egy utasításnak az absztrakt programból.

3.1 A kód elhelyezése

Minden a számítást végző taszk azonos kóddal rendelkezik. A kód szerint tudnia kell minden taszknak, hogy melyik más taszknak kell elküldenie a részeredményeit, illetve honnan kell fogadni adatokat.

[1] Először a sorok, majd oszlopok száma, végül ezeket követi ennyi darab egész szám.

Ez a `gyerek.cpp` fájlban található. A szülő forráskódja a `szulo.cpp` állományban található meg.

4. A program beüzemelése és használata

A program forrása egy `szulo.cpp`, `gyerek.cpp`, `Makefile.aimk`[2] és egy `input.txt` fájl. A program lefordításához követelmény a PVM fejlesztési csomag telepítése a rendszerre.

1. a program főkönyvtárába kell lépni
2. egyszer ki kell adni az „*aimk links*” parancsot, ami a PVM virtuális gép rendszerébe regisztrálja a lefordítandó programokat
3. a fordítást az „*aimk*” paranccsal lehet lefuttatni

A lefordított program futtatásához PVM virtuális gépre van szükség. A PVM virtuális gép elindulása után a „*spawn -> szulo*” parancs kiadásával futtathatjuk le a programot.

A programnak parancssori argumentum formában kell megadni az input állomány nevét. Amennyiben elmulasztjuk, a program a szabványos hibacsatornára hibaüzenetet ír ki és leáll.

5. Futtatási eredmények

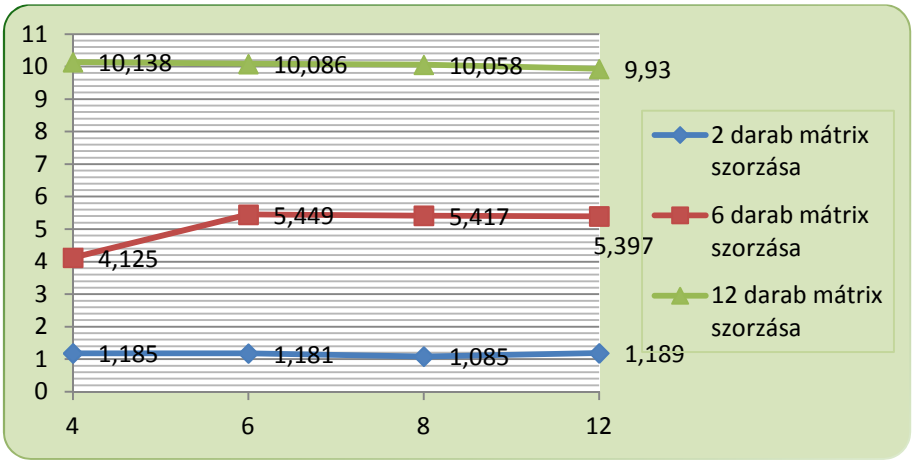
A programot a funkcionális tesztelés után hatékonyságvizsgálatnak vetjük alá. Az első mérés eredményét az alábbi táblázat tartalmazza:

Mátrixok száma:	2	3	4	5	6	7	**8**	**9**	*10*	**11**	**12**
egy processzoron:	7.286s	17.422s	18.599s	26.018s	29.278s	34.423s	37.195s	45.049s	47.332s	55.669s	56.764s
tizenkét processzoron:	14.441s	21.139s	23.562s	29.882s	35.431s	37.194s	36.491s	38.768s	47.439s	51.505s	55.547s

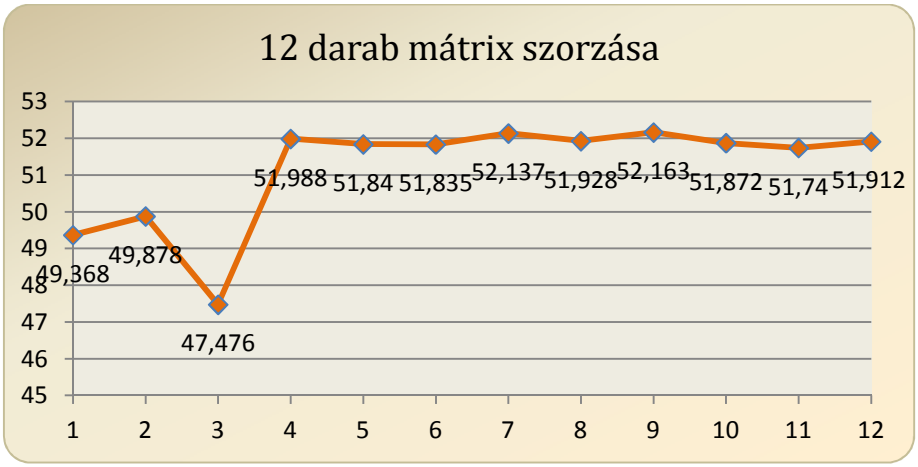
Ezt követően két különböző módon is mérjük a program futásának idejét. Az első módszerrel a processzorok számát növeljük meg, másodikkal a műveletek számát. Ezeknek az eredményét a következő oldal tartalmazza.

[2] A `Makefile.aimk` futtatható program egyszerű elkészítéséhez használható.

1. ábra átlagos méretű mátrixokkal
függőleges tengely: idő(sec), *vízszintes tengely:* processzorok száma



2. ábra közepes[3] méretű mátrixokkal
függőleges tengely: idő(sec), *vízszintes tengely:* processzorok száma



3. ábra nagy[4] méretű mátrixokkal
függőleges tengely: idő(sec), *vízszintes tengely:* processzorok száma

[3] ~500×500-as.
[4] ~1000×1000-es.