Párhuzamos programok implementációja PVM/cluster környezetben - mérési eredmények

Balázs Gábor

Tartalomjegyzék

1. Legnagyobb közös osztó

1.1. A feladat leírása

Adott n db természetes szám. A feladat, hogy ezek közül az első i ($i \in [1..n]$) db legnagyobb közös osztóját meghatározzuk. Ha $n = 2$, akkor nincs nehéz dolgunk, mivel két szám legnagyobb közös osztójának meghatározása ismert: a nagyobb számból vonjuk ki a kisebbet, és ezt addig ismételjük, amíg a két szám egyenlô nem lesz. Tekintsük most ezt az lnko műveletet ismertnek. Ezen felül szükségünk lesz még az lnko: $\mathbf{N} \times \mathbf{N} \longmapsto \mathbf{N}$ művelet asszociatív tulajdonságára is (a kommutativitását most nem használjuk ki). Ezek alapján a feladat visszavezethető az asszociatív művelet kiszámításának tételére. A leírás megkönnyítése érdekében vezessük be az $f : \mathbf{N}^* \longmapsto \mathbf{N}$ függvényt, ami az lnko művelet egyszeri vagy ismételt alkalmazásának felel majd meg.

1.2. Az eredeti feladat specifikációja

$$\begin{array}{llclll} A = & G & \times & G & , \text{ahol } G = \text{vektor } ([1..n], \mathbf{N}) \ , n \geq 1 \\ & g & & a & \\ B = & G & & & \\ & a' & & & \end{array}$$

$$(a = a') \in INIT_{a'}$$
$$\uparrow\hookrightarrow FP_{a'}$$
$$FP_{a'} \Rightarrow (a = a' \wedge \forall i \in [1..n] : g(i) = f(\ll a_i, \ldots, a_1 \gg))$$

1.3. A finomított feladat specifikációja

$$\begin{array}{lccccccccc} A' = & G & \times & G & \times & GS & \times & K & \times & T \\ & g & & a & & gs & & k & & t \end{array}$$

$$\begin{array}{ll} G & = \text{vektor } ([1..n], \mathbf{N}) \\ GS & = \text{vektor } ([1..n, 0..(\lceil \log_2 n \rceil))], \mathbf{N}) \\ K & = \text{vektor } ([1..n], \mathbf{N}_0) \\ T & = \text{vektor } ([1..n], \mathbf{N}_0), n \geq 1 \end{array}$$

Legyen a $h : G \times [1..n] \times \mathbf{N}_0 \longmapsto \mathbf{N}$ parciális függvény, melyre

$$h(a, i, k) = \begin{cases} f(\ll a_i, \ldots, a_1 \gg), & \text{ha } i - 2^k + 1 \leq 1 \\ f(\ll a_i, \ldots, a_{(i-2^k+1)} \gg), & \text{ha } i - 2^k + 1 \geq 1 \end{cases}$$

$$(a = a') \in INIT_{a'}$$
$$\uparrow\hookrightarrow FP_{a'}$$
$$FP_{a'} \Rightarrow \forall i \in [1..n] : (k(i) = \lceil \log_2(n) \rceil) \wedge (g(i) = gs(i, \lceil \log_2(n) \rceil))$$
$$inv_{a'}(\forall i \in [1..n] : k(i) \leq \lceil \log_2(n) \rceil \wedge \forall k : k \leq k(i) : gs(i,k) = h(a,i,k))$$
$$inv_{a'}(\forall i \in [1..n] : t(i) = 2^{k(i)})$$
$$inv_{a'}(a = a')$$

Az eredeti tételben szereplő finomított feladattól eltérően itt a teljes gs mátrix feltöltését megköveteljük. Úgylehet, hogy ez fölösleges plusz feltételnek látszik, azonban ez a végeredmény befolyásolása nélkül jelentősen leegyszerüsíti a megoldó programot, és így könnyebbé válik az implementáció is.
Ez a feladat egyértelműen finomítása az asszociatív művelet tételében megadott finomított feladatnak. Mivel a feladatfinomítás tranzitív tulajdonságú, ezért ez finomítása az eredeit feladatnak is.

1.4. A megoldás

1.5. Az absztrakt program

$$s_0 : \sqcup_{i=[1..n]} gs(i,0), t(i), k(i) := f(\ll a_i \gg), 1, 0$$

$S : (s_0, \{$
$\quad \sqcup_{i=[1..n]} gs(i, k(i)+1), t(i), k(i) :=$
$\quad f(\ll gs(i,k(i)), gs(i-t(i), k(i)) \gg), 2*t(i), k(i)+1,$
$\quad$ ha $(i - t(i) \geq 1) \wedge (k(i-t(i)) \geq k(i))$

$\quad \sqcup_{i=[1..n]} gs(i, k(i)+1), t(i), k(i) :=$
$\quad gs(i,k(i)), 2*t(i), k(i)+1,$
$\quad$ ha $(i - t(i) < 1)$

$\quad \sqcup_{i=[1..n]} g(i) := gs(i,k(i))$, ha $(k(i) = \lceil \log_2(n) \rceil)$
$\})$

Az első értékadáscsoport az asszociatív művelet kiszámításának I. tételében szereplő absztrakt program feltételes értékadásainak egy összevont változata. Ezt az egyszerűsítést a teljesen kitöltött mátrix miatt tehettük meg. Ezt a kritériumot a második értékadáscsoport biztosítja. A harmadik értékadáscsoport már csak a végleges értékeket másolja át a g vektorba.

1.6. Implementáció

Az implementációt elosztott rendszeren valósítjuk meg. A gs mátrix és a k illetve t vektorok i. ($i \in [1..n]$) eleméhez egy taszk tartozik.
Ezeken kívül lesz még egy fő taszk, melynek feladata, hogy a taszkokat elindítsa és ellássa őket a szükséges információkkal. Ezek az adatok az input adatok, az input méret, a többi taszk azonosítója és a taszk helyzete. Az input méret és a taszkok száma megegyezik, és minden egyes taszk egy input adatot kap.
Az absztrakt programnál minden értékadás „elkérte" a számára szükséges adatot. Ez az implementációban megfordul. Ha egy taszk (i.) előállít egy adatot ($k+1$.-et), akkor elküldi azt annak a taszknak ($i+2^k$.-nak), amelynek szüksége lesz rá.
Mivel a taszkok aszinkron módon végzik a feladatukat, előfordulhat az az eset, amikor egy lassabb taszknál sok adat torlódik fel. Mivel az adatok nem biztos, hogy abban a sorrendben érkeznek, mint azt a fogadó taszk várja, így szükséges az elküldött csomagok azonosítása. Minden csomagot a küldő taszk azonosítója és egy szám reprezentál, mely megadja, hogy az adat mely szinten került kiszámításra. Így a fogadó már képes kiválasztani a neki kellő csomagot.

1.7. Teszteredmények

adatszám	1 proc.	5 proc.	10 proc.	15 proc.	20 proc.
10	1m	1m	1m	1m	
50	10m	2m	1m	1m	
100	27m	6m	4m	3m	
250	1p36m	18m	10m	9m	
500	3p47m	42m	24m	17m	
750	NA	1p16m	38m	19m	
1000	NA	2p21m	1p	36m	

2. Skaláris szorzás adott vektorral

2.1. A feladat leírása

Adott egy $v \in \mathbf{Z}^N$ -es ($N \in \mathbf{N}$) vektor. A feladat, hogy határozzuk meg ennek a vektornak és további M db $\mathbf{Z}^N$ - es vektornak a skaláris szorzatát. Ezt az input sorozatot $D =< d_0, d_1, \ldots, d_M >$ - el jelöljük.
A skaláris szorzás több elemi műveletből tevődik össze, és ezek szétbontása lehetőséget ad a párhuzamosításra.
Az eredeti műveletet jelölje $F : \mathbf{V} \longmapsto \mathbf{Z}$, ahol $a \in \mathbf{Z}^N$ - re $F(a) = \sum_{i=1}^{N} (v_i * a_i)$. Továbbá vezessük be az $F(D) =< F(d_1), F(d_2), \ldots, F(d_M) >$ jelölést is.

2.2. Az eredeti feladat specifikációja

$$
\begin{array}{rclllllll}
A & = & \underset{x_0}{Ch(V)} & \times & \underset{\overline{x_0}}{Ch(V)} & \times & \underset{x_{N+1}}{Ch(\mathbf{Z})} & \times & \underset{\overline{x_{N+1}}}{Ch(\mathbf{Z})} \quad \text{, ahol } V = \text{vekt}([1..N], \mathbf{Z}) \\
B & = & \underset{x_0'}{Ch(V)} & \times & \underset{\overline{x_0}'}{Ch(V)} & \times & \underset{x_{N+1}'}{Ch(\mathbf{Z})} & \times & \underset{\overline{x_{N+1}}'}{Ch(\mathbf{Z})}
\end{array}
$$

$$Q ::= (x_0 = \overline{x_0} = x_0' = \overline{x_0}' = D \wedge x_{N+1} = \overline{x_{N+1}} = x'_{N+1} = \overline{x_{N+1}}' =<>)$$

$$Q \in INIT_{\underbrace{x_0' \overline{x_0}' x'_{N+1} \overline{x_{N+1}}'}_{h}}$$

$$FP_h \Rightarrow \overline{x_{N+1}} = F(\overline{x_0}') = F(D)$$

$$Q \hookrightarrow_h FP$$

$$(\overline{x_0} = \overline{x_0}' = D) \in inv_h$$

2.3. A finomított feladat specifikációja

Bontsuk fel az F függvényt a következő módon. Legyenek $f_0 : \mathbf{V} \longmapsto \mathbf{V} \times \mathbf{Z}$, $\forall i \in [1..N-1] : f_i : \mathbf{V} \times \mathbf{Z} \longmapsto \mathbf{V} \times \mathbf{Z}$ és $f_N : \mathbf{V} \times \mathbf{Z} \longmapsto \mathbf{Z}$, ahol $a \in \mathbf{Z}^N$ - re : $f_0(a) = (a, 0)$, $\forall i \in [1..N-1]$: $f_i(a, sum) = (a, sum + v_i * a_i)$ és $f_N(a, sum) = (sum + v_N * a_N)$. Ekkor be kell látnunk, hogy $F = f_N \circ f_{N-1} \circ \ldots \circ f_1 \circ f_0$. Legyen $f^i := \circ_{j=0}^{i} f_j$. Tehát az állítás, hogy $F = f^N$. Legyen $a \in \mathbf{Z}^N$. Ekkor $f^0(a) = f_0(a) = (a, 0)$. Tegyük fel, hogy egy rögzített $i \in [0..N-1]$ - re $f^i(a) = (a, \sum_{j=1}^{i} v_j * a_j)$. Ekkor $f^{i+1} = f_{i+1} \circ f^i = f_{i+1}(a, \sum_{j=1}^{i} v_j * a_j) = (a, \sum_{j=1}^{i+1} v_j * a_j)$. Ekkor $f^N = f_N \circ f^{N-1} = f_N(a, \sum_{j=1}^{N-1} v_j * a_j) = \sum_{j=1}^{N} v_j * a_j = F(a)$.
Ezen kívül bővítsük ki az állapotteret további ($x_i, i \in [1..N]$) $Ch((\mathbf{V}, \mathbf{Z}))$ típusú adatcsatornaváltozókkal és az ezekhez tartozó történetváltozókkal.

$$
\begin{array}{lllllllll}
A = & Ch(V)\times & Ch(V)\times \ldots & \times & Ch((\mathbf{V},\mathbf{Z}))\times & Ch((\mathbf{V},\mathbf{Z}))\times \ldots & \times & Ch(\mathbf{Z})\times & Ch(\mathbf{Z}) \\
 & x_0 & \overline{x_0} & & x_i & \overline{x_i} & & x_{N+1} & \overline{x_{N+1}} \\
B = & Ch(V)\times & Ch(V)\times \ldots & \times & Ch((\mathbf{V},\mathbf{Z}))\times & Ch((\mathbf{V},\mathbf{Z}))\times \ldots & \times & Ch(\mathbf{Z})\times & Ch(\mathbf{Z}) \\
 & x'_0 & \overline{x_0}' & & x'_i & \overline{x_i}' & & x'_{N+1} & \overline{x_{N+1}}'
\end{array}
$$

$$Q ::= (x_0 = \overline{x_0} = x'_0 = \overline{x_0}' = D \wedge x_{N+1} = \overline{x_{N+1}} = x'_{N+1} = \overline{x_{N+1}}' =<>)$$

$$Q \in INIT_{\underbrace{x'_0\overline{x_0}'x'_{N+1}\overline{x_{N+1}}'}_{h}}$$

$$(\forall i \in [0..N] : f_i(\overline{x_i} - x_i) = \overline{x_{i+1}}) \in inv_h$$

$$FP_h \Rightarrow \forall i \in [0..N] : x_i =<>$$

$$Q \hookrightarrow_h FP$$

$$(\overline{x_0} = \overline{x_0}' = D) \in inv_h$$

2.4. A megoldás

2.5. Az absztrakt program

$S : (\|_{i=1}^{N}\ x_i :=<>,\ \{\ \sqcup_{i=0}^{N} x_i, x_{i+1} := lorem(x_i), hiext(x_{i+1}, f_i(x_i.lov))$, ha $x \neq <>\})$, azaz
$S : (\|_{i=1}^{N}\ x_i :=<>$, $\{\ x_0, x_1 := lorem(x_0), x_1; (x_0.lov, 0)$, ha $x_0 \neq <>$
$\sqcup_{i=1}^{N-1} x_i, x_{i+1} := lorem(x_i), x_{i+1}; (x_i.lov_1, x_i.lov_2 + v_i * a_i)$, ha $x_i \neq <>$
$\sqcup x_N, x_{N+1} := lorem(x_N), x_{N+1}; (x_N.lov_2 + v_N * a_N)$, ha $x_N \neq <>\ \})$

Mivel a megoldás az adatcsatorna tételen alapszik, ezért helyességét külön nem kell belátnunk.

2.6. Implementáció

Az implementációt olyan elosztott rendszeren valósítjuk meg, melyen a szorzás művelet költsége nagy, a kommunikációé pedig kicsi (mivel a Grid tesztkörnyezet nem ilyen, ezért a szorzásművelet lassításával szimuláltuk ezt). A rögzített vektort és az input adatokat egy file-ból olvassuk, ami ezeken kívül tartalmazza még a vektorok méretét és az input adatok számát is.
A programban a fő taszk játsza az f_0 szerepét, és ezen kívül elindítja az altaszkokat, amiket ellát a működésükhöz szükséges információkkal.
Az altaszkok felelnek meg az $f_1, \ldots f_N$ függvényeknek. Azt, hogy egy altaszk f_i $(i \in [1..N-1])$ -nek, vagy f_N - nek felel-e meg egy elágazás dönti el a programban.
Magukat az adatcsatornákat az altaszkokhoz tartozó bufferek reprezentálják, melyeket a pvm környezet biztosít.

Az input adatok számára azért van szükség, mert ellenben az absztrakt programmal a valódi programnak terminálnia kell (nem absztrakt értelemben). Az altaszkok pedig így döntik el, hogy az összes feldolgozandó vektor áthaladt-e már rajtuk.

2.7. Teszteredmények

adatszám	1 proc.	5 proc.	10 proc.	15 proc.	20 proc.
50	13m	7m	4m	2m	
100	25m	10m	6m	4m	
200	46m	19m	9m	7m	
500	1p50m	43m	21m	13m	
1000	3p36m	1p25m	41m	26m	
2000	4p9m	2p43m	1p21m	46m	
5000	17p50m	7p0m	3p16m	2p11m	

A tesztelés 100 hosszúságú vektorokkal és -500 - 500 közötti adatelemekkel történt. A tesztadatokat véletlenszerűen generáltuk.