Jan Vybíral, xvybir05

Dokumentace ke 3. projektu do PRL: Implementace algoritmu “Mesh Multiplication”

1. Princip algoritmu a analýza složitosti

```
Algoritmus
procedure MESH MULT(A, B, C)
for i=1 to m do in parallel
  for j=1 to k do in parallel
    c_ij = 0
    while P(i,j) receives inputs a,b do
        c_ij = c_ij + (a * b)
        if i<m then send b to P(i+1, j)
        if j<k then send a to P(i, j+1)
    endwhile
  endfor
endfor
```

Pro vynásobení dvou matic $A[m,n]$ a $B[n,k]$ je použito pole procesorů $P[m,k]$ o rozměrech výstupní matice.

Prvky $a_{m,1}$ a $b_{1,k}$ potřebují $m+k+n-2$ kroků, aby se dostaly k poslednímu procesoru $P_{m,k}$, takže celková časová složitost algoritmu je lineárně závislá na rozměru výstupní matice:

$$t(n) = O(n)$$

Počet procesorů potřebných pro řešení úlohy je $m * k$, tedy kvadraticky závislý na rozměru výstupní matice:

$$p(n) = O(n^2)$$

Celková cena implementace je tedy

$$c(n) = p(n) * t(n) = O(n^2) * O(n) = O(n^3)$$

Což není optimální.

2. Implementace a komunikační protokol

V následujícím textu se předpokládá indexování od 0.

Procesor $P_{0,0}$ nejdříve načte obě matice ze vstupních souborů, uloží si je a s pomocí broadcastu pošle všem ostatním procesorům informaci o tom, jestli byla všechna data úspěšně načtena a jsou v korektním formátu. Pokud ne, všechny procesory se ukončí s návratovou hodnotou -1. Procesor $P_{0,0}$ poté opět s pomocí broadcastu zašle všem ostatním procesorům informace o rozměrech vstupních matic. Na základě těchto hodnot si každý procesor spočítá svoji pozici v poli procesorů. Procesor $P_{0,0}$ následně pošle každému procesoru $P_{0,j}$, kde $j = 1 \dots k$, hodnoty z j-tého sloupce matice B, a každému procesoru $P_{i,0}$, kde $i = 1 \dots m$, hodnoty z m-tého řádku matice A. Tyto procesory si obdržené hodnoty uloží do polí, ze kterých budou později načítat své vstupy. Procesor $P_{0,0}$ si také uloží hodnoty z prvního řádku matice A a prvního sloupce matice B do svých vstupních polí. V dalším kroku všechny procesory vynulují svůj registr c.

Samotný výpočet provádí každý procesor v cyklu o n krocích. V každém kroku cyklu se provede toto:

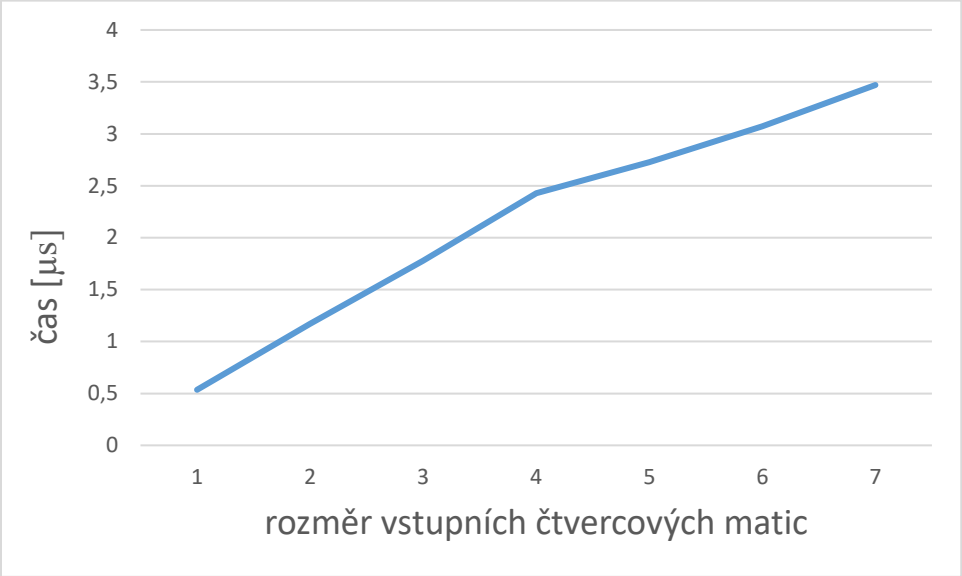
Každý procesor $P_{i,j}$ obdrží jednu hodnotu a z i-tého řádku matice A a jednu hodnotu b z j-tého sloupce matice B. Procesory v prvním řádku a prvním sloupci si tyto hodnoty berou ze svých vstupních polí, kam byly při inicializaci uloženy. Ostatní procesory obdrží hodnotu a od svého levého souseda a hodnotu b od svého horního souseda. Každý procesor tyto dvě obdržené hodnoty mezi sebou vynásobí a výsledek přičte k hodnotě svého registru c. Následně každý procesor kromě procesorů v posledním řádku a posledním sloupci zašle hodnotu a svému pravému sousedovi a hodnotu b svému spodnímu sousedovi.

Poté, co se toto provede n-krát, obsahují registry c všech procesorů hodnoty korespondujících prvků výstupní matice. Všechny procesory kromě procesoru $P_{0,0}$ zašlou hodnotu svého registru c procesoru $P_{0,0}$, který vypíše výslednou matici na standardní výstup.

3. Experimentální ověření časové složitosti

Čas potřebný k vykonání implementovaného algoritmu jsem měřil na serveru *merlin* za použití nástrojů obsažených v hlavičkovém souboru <*chrono*>, které mimo jiné umožňují měřit čas uplynulý mezi libovolnými dvěma místy v programu v mikrosekundách. Do měření nebylo zahrnuto: inicializace knihovny *OpenMPI,* načítání vstupů ze souborů, počáteční rozesílání řádků/sloupců vstupních matic krajním procesorům, závěrečné posílání výsledků od všech procesorů nultému procesoru a tisknutí výstupu. Testování probíhalo pro jednoduchost pouze na čtvercových maticích o rozměrech 1 až 7. Testování větších matic nebylo bohužel na serveru *merlin* možné. Pro každou velikost vstupu bylo provedeno 10 měření a takto získané hodnoty byly poté zprůměrovány.

Následující graf znázorňuje celkový výsledek testování:



Z grafu je vidět, že čas potřebný pro provedení výpočtu neroste v závislosti na rozměrech vstupních čtvercových matic rychleji než lineárně, což odpovídá teoretické časové složitosti uvedené výše.

4. Závěr

Implementoval jsem paralelní algoritmus *Mesh Multiplication* pro násobení matic a provedl měření jeho časové složitosti. Zjistil jsem, že výsledek odpovídá očekávané lineární složitosti.