Mesh Multiplication
Dokumentácia k projektu č. 3

Peter Lacko
xlacko06@stud.fit.vutbr.cz

27. dubna 2015

Cieľom tohoto projektu bolo implementovať algoritmus Mesh Multiplication vykonávajúci násobenie dvoch matíc v paralelnom prostredí. Algoritmus bol implementovaný v jazyku C/C++ s využitím knižnice *OpenMPI*[1]. Vstupom program sú dve matice ľubovoľných rozmerov, $mat1$ a $mat2$ (rešpektujúc $|cols(mat1)| = |rows(mat2)|$) uložených v súboroch `mat1` a `mat2`, výstupom je ich súčin $mat1 \cdot mat2$.

1 Popis a analýza algoritmu Mesh Multiplication

Algoritmus pracuje nasledovne: Nech matica $mat1$ má rozmery $m \times n$ (m = počet riadkov, n = počet stĺpcov), a matica $mat2$ rozmery $n \times k$. Výsledná matica matica $mat3$ má potom rozmery $m \times k$ a na výpočet je potreba $m \cdot k$ procesorov umiestnených v mriežke $m \times k$. Procesory $P_{0,j}$ pre $j = 0 \ldots k-1$ obsahujú na počiatku jednotlivé stĺpce matice $mat2$ a procesory $P_{i,0}$ pre $i = 0 \ldots m-1$ jednotlivé riadky matice $mat1$. Hodnota v každom procesore je na počiatku inicializovaná na 0, v každom cykle si každý procesor $P_{i,j}$ prečíta hodnoty i_1 a i_2 prijaté z procesorov $P_{i-1,j}$ resp. $P_{i,j-1}$ (okrem procesorov v najľavejšom stĺpci a v hornom riadku spomenutých vyššie), aktualizuje svoju hodnotu ako pričítaním súčinu $i_1 * i_2$ k aktuálnej hodnote. Ak sa procesor nenachádza v poslednom stĺpci, resp. riadku, pošle hodnoty i_1 a i_2 procesorom $P_{i+1,j}$ resp $P_{i,j+1}$. Sekvenčný diagram zasielania správ medzi jednotlivými procesmi je zobrazený na obr. 1.

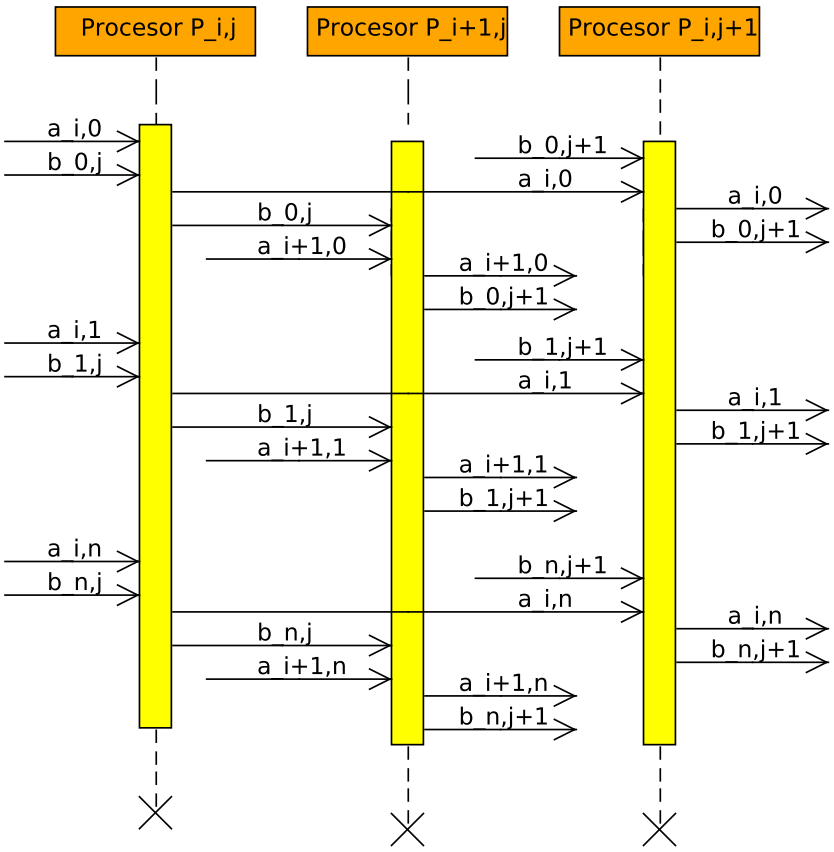
Procesor $P_{m,k}$ začne vykonávať svoju činnosť v najhoršom prípade po $m+k-2$ cykloch a spracováva n hodnôt. Za predpokladu že $m \leq n$ a $k \leq n$ je jeho časová zložitosť $t(n) = O(n)$, a keďže $p(n) = O(n^2)$, cena $c(n) = t(n) \cdot p(n) = O(n^3)$ čo odpovedá cene naivnému sekvenčného algoritmu. Algoritmus však nie je optimálny keďže nie je známy optimálny sekvenčný algoritmus. Vzhľadom k tomu že dáta sa medzi procesmi zasielajú, a teda nijak neduplikujú, ja priestorová zložitosť algoritmu $O(n)$.

2 Implementácia algoritmu

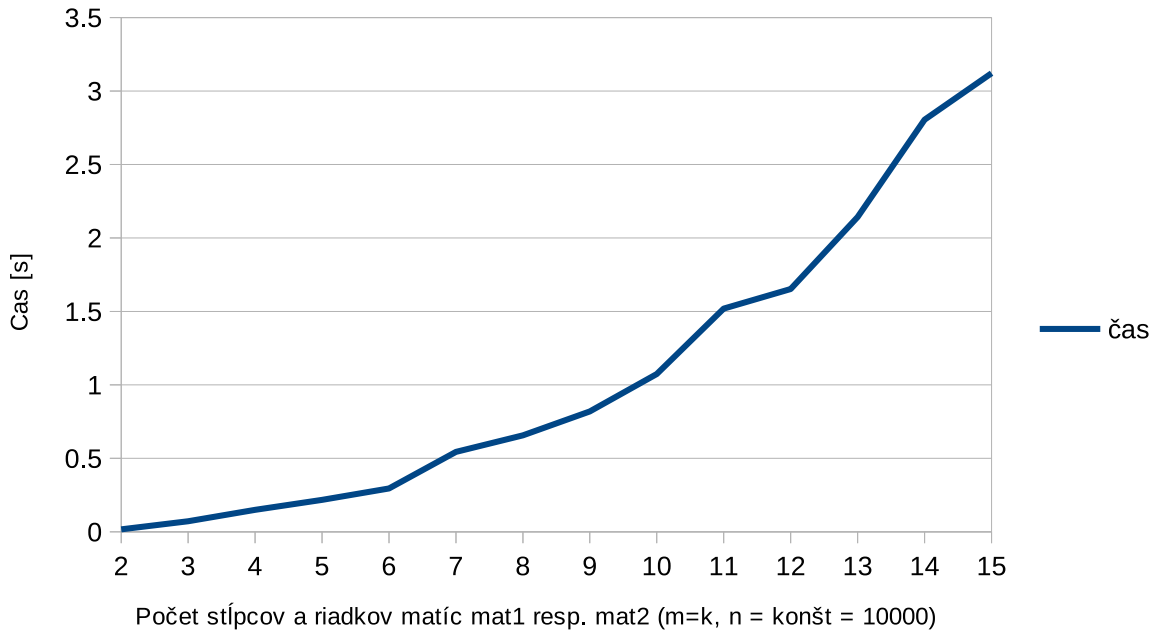
Činnosť programu je implementačne rozdelená do troch hlavných častí: spracovanie vstupných súborov a distribúcia hodnôt, implementácia algoritmu a zaslanie výsledných hodnôt procesoru P_0 a ich výpis.

Spracovanie vstupných súborov a distribúcia hodnôt. Procesor $P_{0,0}$ načíta do pamäte obsahy súborov mat1 a mat2, následne zašle procesorom $P_{1,0}$ a $P_{0,1}$ dimenzie výslednej matice m a k a počet spracovávaných hodnôt n, ktoré ich obdobne zašlú procesorom v dalšom rade, resp. stĺpci, atd. Procesor $P_{0,0}$ ďalej rozošle riadky matice mat1 procesorom $P_{i,0}$ pre $i = 0 \ldots m-1$ a stĺpce matice mat2 procesorom $P_{0,j}$ pre $j = 0 \ldots k-1$.

[1] OpenMPI implementuje štandard/špecifikáciu MPI (Message Passing Interface), popisujúci komunikáciu procesorov v paralelnom prostredí pomocou zasielania správ medzi nimi.



Obrázek 1: Sekvenčný diagram komunikácie procesov.



Obrázek 2: Dĺžka trvania algoritmu pri konštantnom $n = 10000$ v závislosti na veľkosti vstupných matíc.

Implementácia algoritmu. Samotný algoritmus je priamočiary a presne odpovedá popisu uvedenému vyššie.

Zaslanie výsledných hodnôt procesoru $P_{0,0}$ a ich výpis. O zaslanie výsledkov procesoru $P_{0,0}$ sa stará funkcia `MPI_Gather()` doslova zhromažďujúca výsledky jednotlivých procesorov a ukladá ich do vopred naalokovaného poľa procesoru $P_{0,0}$ ktorý ich následne vypíše na štandardný výstup.

3 Spôsob testovania a dosiahnuté výsledky

Testovanie rýchlosti algoritmu prebiehalo na lokálnom PC a to pre konštantnú hodnotu $n = 10000$ a rôzne veľkosti $m = k = 2 \dots 15$. Cieľom merania bolo overiť lineárnu časovú zložitosť algoritmu. Ako je patrné z grafu 1, čas potrebný na výpočet sa zvyšuje takmer lineárne so zvyšujúcim sa počtom vstupov. Miernu nelinearitu je možné vysvetliť narastajúcou réžiou spojenou s komunikáciou procesov a taktiež tým, že procesy nebežia v paralelnom prostredí.

Výsledný čas je aritmetickým priemerom z desiatich behov. Čas bol meraný pomocou funkcie MPI_Wtime(), vracajúcej skutočný uplynutý čas na procesore. Meranie je vykonávané procesorom $P_{0,0}$ a vyjadruje čistý čas trvania algoritmu, tj. od započatia výpočtu procesorom $P_{0,0}$ (po distribúcii hodnôt) po ukončenie posledného výpočtu procesoru $P_{m-1,k-1}$.

4 Zhodnotenie dosiahnutych výsledkov

Výstupom projektu je program schopný násobiť matice obsahujúce rádovo desiatky tisíc prvkov. Počet riadkov resp. stĺpcov (a teda aj veľkosť výslednej matice) je však obmedzená dostupnými zdrojmi – počtom dostupných procesorov/procesov.

Overili sme taktiež, že dosiahnutá časová zložitosť algoritmu takmer odpovedá zložitoti teoretickej, pričom mierna nalinearita je spôsobená zvýšenou réžiou súvisiacou s narastajúcou veľkoťou matíc a faktom, že procesy nebežia v paralelnom prostredí.

Reference

[1] Selim G. Akl. *Parallel Sorting Algorithms.* Academic Press, Inc., 1990.