# Implementace algoritmu "Mesh Multiplication"

## PRL: 3. Projekt

*23. 4. 2017*

**Jakub Pastuszek, xpastu00@stud.fit.vutbr.cz**

## Analýza algoritmu

Algoritmus „Mesh Multiplication“ pracuje s polem procesorů $p(n) = M * K$, pro vstupní matice $A$, $B$ o rozměrech $A = M \times N$ a $B = N \times K$. Číslo $M$ určuje počet řádků první matice a číslo $K$ počet sloupců druhé matice, zatímco číslo $N$ udává počet sloupců první matice a počet řádků matice druhé. Z $N$ vyplývá fakt, že matice tuto hodnotu musejí mít stejnou. Mimo jiné výsledná matice má počet řádků roven $M$ a počet sloupců roven $K$.

Optimální sekvenční algoritmus neexistuje, ale ví se, že jeho složitost nemůže být lepší než $O(n^2)$ a je určitě lepší než $O(n^3)$. Žádný sekvenční algoritmus nemá lepší časovou složitost, než je $O(n^2)$, a to z důvodu, že je potřeba projít všechny prvky každé z matic alespoň jednou. Složitost algoritmu „Mesh Multiplication“ je udávaná jako $O(n^3)$ a toto tvrzení bude prověřeno v kapitole Experimenty.

Nejprve je nutné jedním procesorem načíst vstupní data (jako distribuční procesor je vhodné zvolit procesor s indexem *„n-1“*, tj. poslední, protože se k němu dostanou data až naposled) a redistribuovat je dalším procesorům. Distribuce je prováděna po sloupcích a řádcích, a to vždy prvním procesorům v řádku a sloupci. Prvky matic $a_{M,1}$ a $b_{1,O}$ potřebují M+K+N-2 kroků, aby se dostaly k poslednímu procesoru $P(M,K)$. Z tohohle počtu kroků plyne lineární asymptotická časová složitost $t(n) = O(n)$.

Cena paralelního algoritmu je dána násobkem prostorové složitosti (počtu procesorů) a časové složitosti. Z prostorové složitosti $p(n) = M * K$ vyplývá asymptotická prostorová složitost algoritmu $p(n) = O(n^2)$. Provedení algoritmu je dostatečně rychlé $t(n) = O(n)$, ale velký počet procesorů celkovou cenu značně zhorší, proto je celková cena $c(n) = O(n^3)$, což není optimální.

## Implementace

Každý procesor obsahuje 3 pracovní registry označené A, B, C. Počáteční hodnota registru C každého procesoru je 0, tato hodnota označuje výslednou hodnotu prvku matice.

Je zvolen jeden distribuční procesor, který načte data z obou souborů a poté všem procesorům rozešle zprávu o počtu řádků ($M$) a sloupců ($K$) výsledné matice a dále počet sloupců první matice (resp. počet řádků matice druhé), hodnotu $N$. Vhodné je takto zvolit procesor s indexem *„n-1“*, tj. poslední, protože se k němu data dostanou až nejpozději ze všech procesorů – po $min(M,K)$ krocích.

Topologie procesorů je 2D mřížka, kde lineární pole procesorů je uspořádáno po sloupcích do mřížky. Každý procesor příjme dvě hodnoty a to jednu do registru A, buďto od distribučního procesoru nebo od levého souseda (chápáno jako procesor s indexem o $K$

menším) a druhou do registru B a to buď od distribučního procesoru nebo od horního souseda (chápáno jako procesor s indexem o 1 menším). Dále se spočte součin těchto přijatých čísel a výsledná hodnota se přičte k hodnotě v registru C.

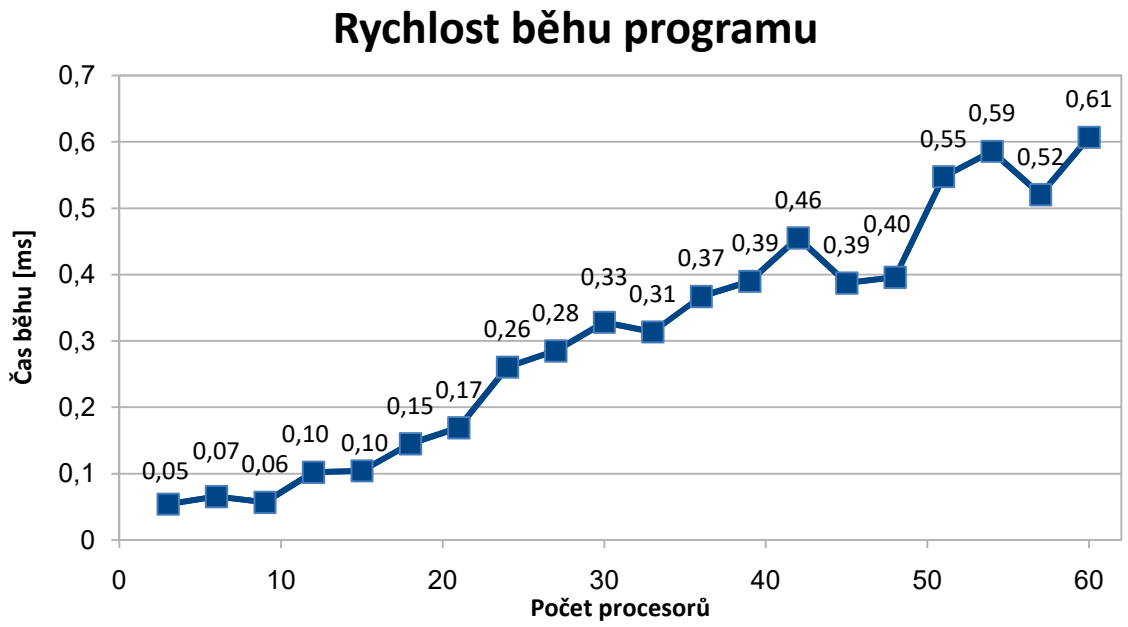
Po přijetí *N* hodnot z obou směrů má každý procesor ve svém registru C výslednou hodnotu prvku matice. Pozice prvku ve výsledné matici je dána pozicí procesoru v dané topologii – 2D mřížce. Dále každý procesor zašle tuto hodnotu procesoru distribučnímu, který tyto hodnoty následně správně vytiskne na výstup.

## Experimenty

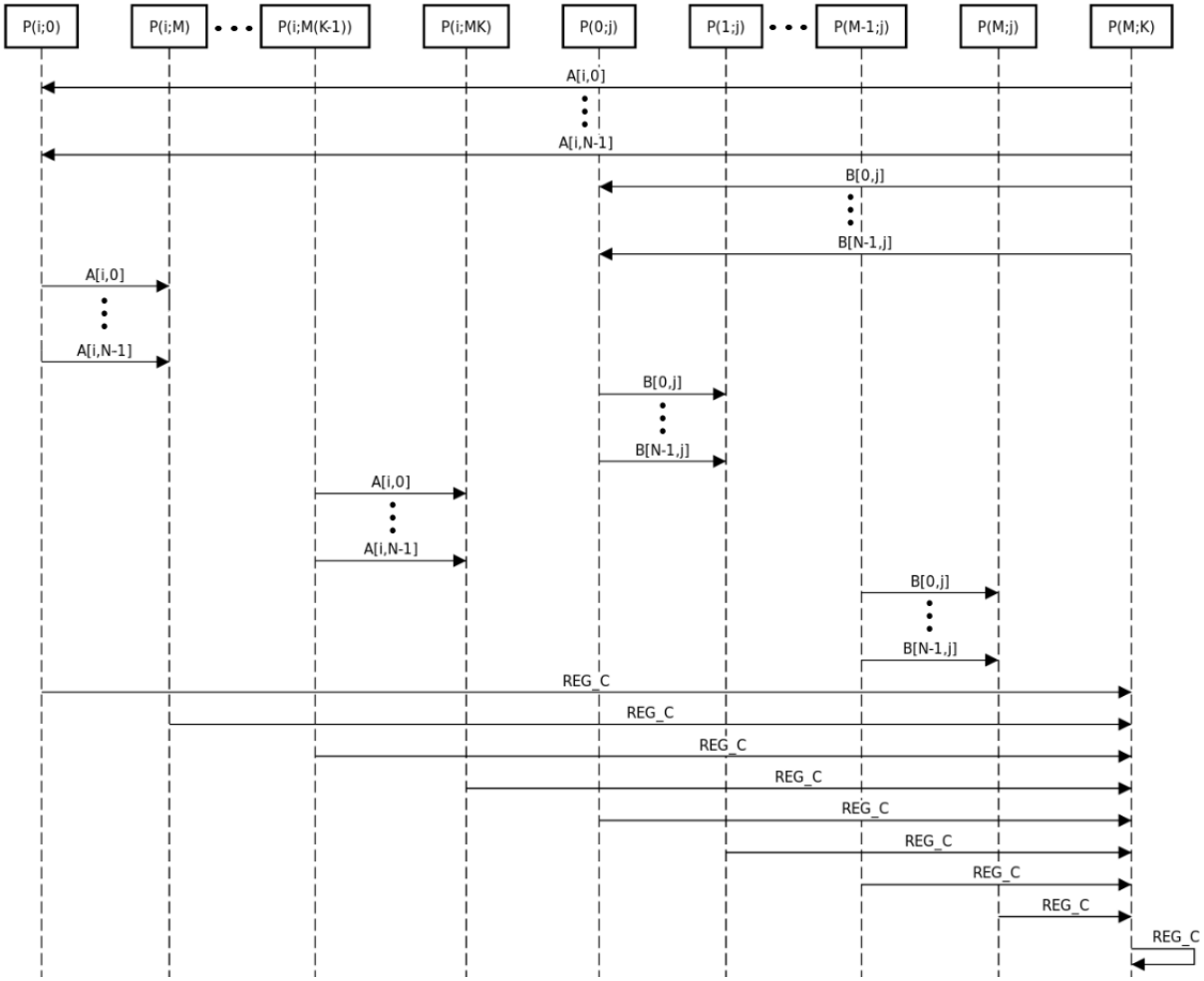
Testování doby běhu programu probíhalo v závislosti na velikosti vstupních matic. Vstupní matice A zůstala po dobu testování neměnná, zatímco velikost matice B (její počet sloupců) byl postupně zvyšován pro otestování doby běhu algoritmu v závislosti na velikosti vstupních dat, zatímco rozměr *N* zůstal po dobu testování stejný. První matice, a to matice A byla fixní o rozměrech *3x3*. Druhá matice B začala na rozměrech *3x1*, kde se postupně zvyšoval počet sloupců až do hodnoty 20. Počty požadovaných procesorů pot byly 3, 6, 9, …, 60 (viz graf). Aby výsledky byly relevantní, byly tyto měření pro každý počet procesorů opakovány 100x a následně zprůměrovány. Měření času bylo provedeno pomocí funkce *„MPI_Wtime“*. Startovací časová značka se nacházela po inicializaci MPI a načtení souborů a koncová byla před finálním výpisem. Před oběma voláními funkce byla vložena bariéra. Výsledky testování jsou vyneseny v grafu níže. Testování probíhalo na lokálním PC s následujícími parametry: CPU – Intel i5-6300HQ@2.30GHz, Mem – 8,00 GB, OS – Xubuntu 16.04 x64.

Nebyly testovány běhy, kdy by byly spuštěny pouze dva nebo méně procesorů, protože v tomto případě by byly výsledky značně ovlivněny latencí zasílaných zpráv.

Potřebný čas pro dokončení úlohy se lineárně zvětšuje s velikostí vstupních matic (počtu procesorů). Hodnoty naměřené při testování jsou v souladu s teoretickou časovou složitostí algoritmu, proto lze označit implementaci za správnou.



## Komunikační protokol




## Závěr

Paměťově je tento algoritmus málo náročný, protože každý procesor si potřebuje pamatovat pouze 3 hodnoty. Úkony jednotlivých procesorů nejsou nikterak složité, procesor provádí pouze násobení a sčítání dvou čísel. Poté již přichází na řadu jen zasílání a příjem další čísel. Otestováním algoritmu na různě velkých vstupních maticích se prokázala teoretická časová složitost algoritmu $t(n) = O(n)$. Celkově je algoritmus velice rychlý, ale v případě větších matic již přichází v potaz velké množství procesorů a s tím spojený velký počet zasílaných zpráv, které obsahují pouze jedno číslo.