Paralelní a distribuované algoritmy
Mesh multiplication

Lukáš Dibďák, xdibda00

1 Rozbor a analýza algoritmu

Popis algoritmu

Algoritmus Mesh multiplication pracuje s mřížkou procesorů **$p(n) = n^2$**. Prvky matic se přivádějí do prvních sloupců a řádků mřížky. Každý procesor obsahuje 3 registry: **A**, **B** a **C**.

A	řádková hodnota (hodnota přijatá z procesoru “vlevo”)
B	sloupcová hodnota (hodnota přijatá z procesoru “zhora”)
C	výsledkový registr

Rozbor algoritmu

- Nechť algoritmus dostane na vstup dvě matice **X(m x k)** a **Y (k x n)**, nad kterými má provést operaci.
- Výsledná matice Z bude o velikosti **m x n**. Celkově je potřeba **m x n** procesorů.
- Na procesory prvního řádku, jejichž počet se rovná m, přivádíme hodnoty z druhé matice **Y** a tyto procesory si tyto hodnoty v každém kroku ukládají do registru **B**
- Na procesory prvního sloupce, jejichž počet se rovná n, přivádíme hodnoty z první matice **X** a tyto procesory si tyto hodnoty v každém kroku ukládají do registru **A**
- Všechny procesory inicializují svůj registr **C** na číslo 0
- Každý procesor (pokud není v posledním sloupci, resp. řádku) odesílá hodnoty svých registrů **A** a **B** procesorům “vpravo”, resp. “dolů”
- V každém kroku, pokud procesor obsahuje hodnoty v obou registrech **A** a **B**, pak tyto hodnoty vynásobí a přičte je k registru **C**

2 Teoretická složitost

- Jak již bylo naznačeno výše, pro operaci nad vstupními maticemi je potřeba m x n procesorů. Jedná se tedy o kvadratickou složitost.
- Složitost samotného výpočtu je následně lineární.
- Celková složitost je exponenciální, což není optimální.

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

3 Implementace

Nejdříve se pomocí příkazů **MPI_Init**, **MPI_Comm_size** a **MPI_Comm_Rank** inicializuje prostředí paralelního procesu.

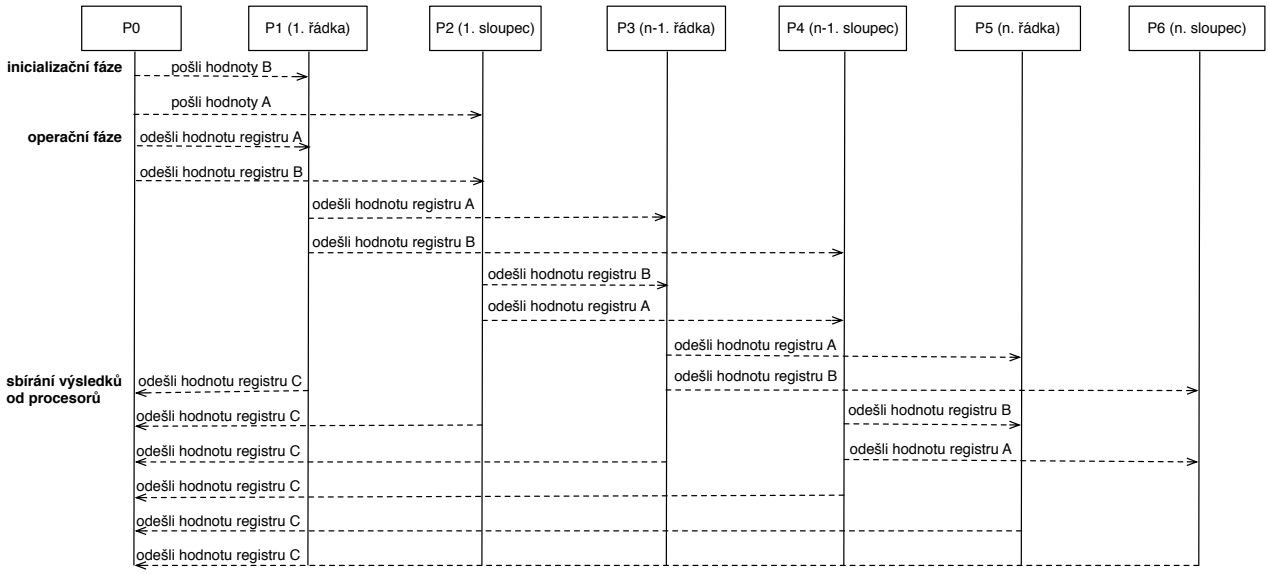
V první části procesor číslo 0 načte matice ze souborů a zvaliduje je. Pokud narazí na nevalidní matici se špatnou syntaxí, pak ukončí program a pošle signál **MPI_Abort** všem ostatním procesorům. Pokud jsou vstupní matice validní, pak připraví velikost výstupní matice, informuje ostatní procesory o této velikosti pomocí funkce **MPI_BCast** procesorům v prvním řádku a prvním sloupci odešle hodnoty.

V druhé části procesory všechny procesory inicializují hodnotu svého registru **C** na inicializační hodnotu 0 a následně provádí dané výpočetní kroky. V každém kroku zkontroluje hodnoty registrů **A** a **B** a pokud nejsou prázdné, pak tyto registry znásobí a přičte tuto hodnotu k registru **C.** Následně, pokud není procesor poslední v dané řadě, přepošle hodnoty dalšímu procesoru.

V momentu, kdy jsou procesory hotové se zpracováním a operaci nad hodnotami, přepošlou své registry **C**, ve kterých se nalézá výsledek, prvnímu procesoru, který je zpracuje a vytvoří konečnou výstupní matici.

4 Experimenty s různě velkými vstupy

5 Komunikační protokol



6 Závěr

Dosažené výsledky hodnotím jako přijatelné. Nelze v reálném prostředí získat výsledky odpovídající teoretické složitosti daného algoritmu. Důvodem tohoto zkreslení může být režie procesorů při provádění programu, zpomalení na serveru, testovací stroj či neoptimálnost implementace algoritmu v programu.