Rozbor a analýza algoritmu

Algoritmus je realizován na mřížce o velikosti $m \times k$ procesorů. Vstupem algoritmu jsou matice A ($m \times n$) a matice B ($n \times k$). Prvky matice A jsou postupně přiváděny do procesorů prvního řádku a prvky matice B jsou přiváděny do procesorů prvního sloupce. Každý procesor $P_{(i, j)}$ tedy obsahuje prvek c_{ij}.

V prvním kroku algoritmus vynásobí prvky a_{11} a b_{11} a výsledek přičte k procesoru $P_{(1, 1)}$. V druhém kroku se prvek a_{11} přesune do procesoru $P_{(1, 2)}$ a vynásobí s prvkem b_{12}, který je přiveden na vstup procesoru a výsledek se přičte k procesoru $P_{(1, 2)}$. Následně procesor $P_{(1, 1)}$ obdrží prvky a_{12} a b_{21}, jenž jsou vynásobeny a přičteny k procesoru $P_{(1, 1)}$ a současně prvek b_{11} je odeslán procesoru $P_{(2, 1)}$. Každý procesor, který se nenachází na pravém okraji mřížky odesílá prvek a sousednímu procesoru $P_{(i + 1, j)}$ a každý procesor, který se nenachází na spodním okraji mřížky odesílá prvek b procesoru $P_{(i, j + 1)}$. Takto algoritmus pokračuje, dokud spolu nejsou vynásobeny prvky a_{nn} a b_{nn} v procesoru $P_{(n, n)}$.

Analýza složitosti

- Aby se prvky a_{m1} a b_{1k} dostaly k poslednímu procesoru $P_{(m, k)}$ je potřeba $m + k + n - 2$ kroků. Z toho vyplývá, že časová složitost **t(n) = O(n)**.
- Prostorová složitost algoritmu **p(n) = n^2**.
- Cena algoritmu **c(n) = t(n) × p(n) = O(n^3)**.

Implementace

Při implementaci jsem vycházel z výše popsaného algoritmu. Aplikace je spuštěna s $m \times k$ procesory, kde m je počet řádků první matice a k je počet sloupců druhé matice. Matice jsou uloženy ve dvou vstupních textových souborech *mat1* a *mat2*. Čísla na řádku jsou oddělena mezerou a řádky jsou odděleny znakem konce řádku. První řádek souboru *mat1* značí počet řádků první matice a první řádek souboru *mat2* značí počet sloupců druhé matice.

Po spuštění aplikace procesor s rankem 0 otevře vstupní soubory a načte matice do paměti a zkontroluje, zda vstupní matice jsou validní. Pokud ne, odešle ostatním procesorům zprávou broadcast signál o chybě a aplikace se ukončí. Dále rozešle příslušné prvky řádkovým a sloupcovým procesorům. Odesílání probíhá synchronně, kdy se počet odeslaných sloupců a řádků vzájemně synchronizuje. Tím je zajištěno, že procesor s rankem 0 obdrží prvky v korektním pořadí. Následně každý řádkový a sloupcový procesor si tyto prvky uloží do vectoru. Nakonec procesor s rankem 0 odešle pomocí zprávy broadcast počet řádků a sloupců vstupních matic ostatním procesorům.

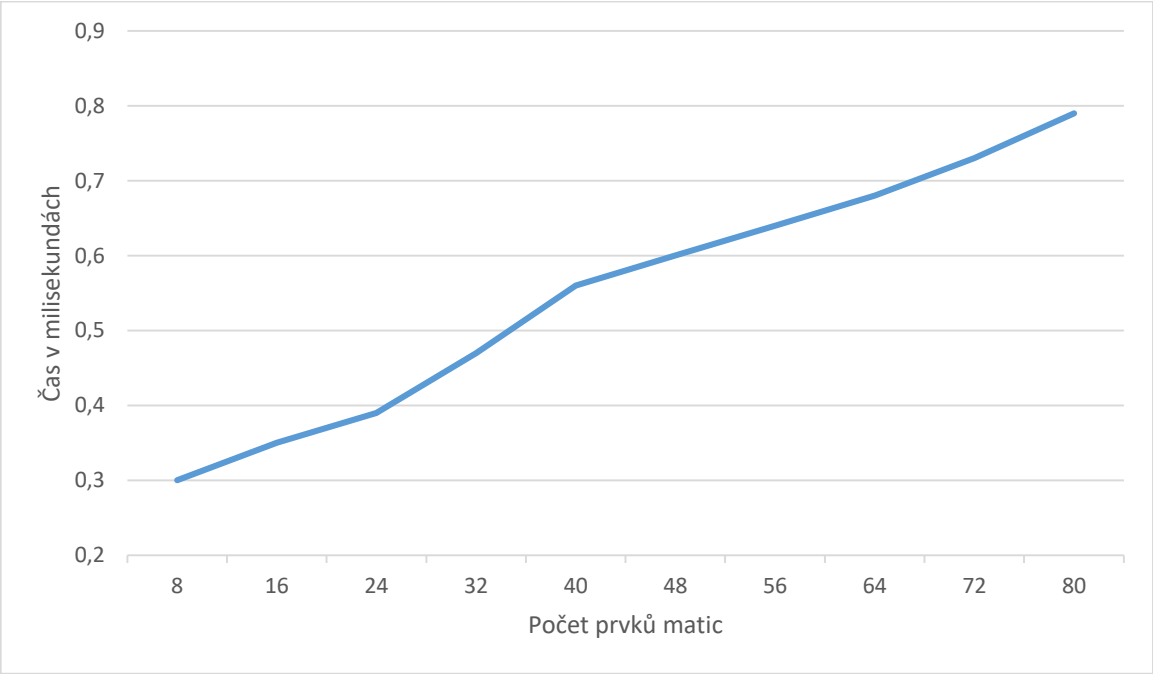
Další část aplikace realizuje samotné násobení matic. Každý procesor inicializuje proměnnou C_{ij} na hodnotu 0. Další kroky jsou provedeny vy smyčce, jejíž počet kroků je roven počtů sloupců první matice. Pokud rank procesoru je 0, načte hodnoty a i b z paměti z vectoru (prvky byly odeslány střídavě jako sloupce a řádky). Sloupcové procesory přijmou hodnotu a od levého sousedního procesoru a hodnotu b z paměti z vectoru. Řádkové procesory přijmou hodnotu b od vrchního sousedního procesoru a hodnotu a z paměti z vectoru. Všechny ostatní procesory hodnoty a i b získávají od sousedních procesorů. Následně se mezi sebou prvky a a

b vynásobí a výsledek je přičten k proměnné C_{ij}. V posledním kroku smyčky jsou hodnoty *a* a *b* odeslány sousedním procesorům.

V posledním kroku jsou výsledné hodnoty odeslány procesoru s rankem 0, který je uloží do paměti a vypíše příslušnou matici na stdout.

Testování

Aplikaci jsem testoval na maticích, kde první matice obsahovala 4 řádky a druhá matice obsahovala 4 sloupce. To znamená, že bylo zapotřebí celkem 16 procesorů. Postupně jsem zvyšoval počet sloupců první matice a počet řádků druhé matice a sledoval čas potřebný k vynásobení matic. Každý test jsem provedl několikrát a časy zprůměroval. Testování proběhlo na školním serveru merlin. Výsledek testování je znázorněn na následujícím grafu:

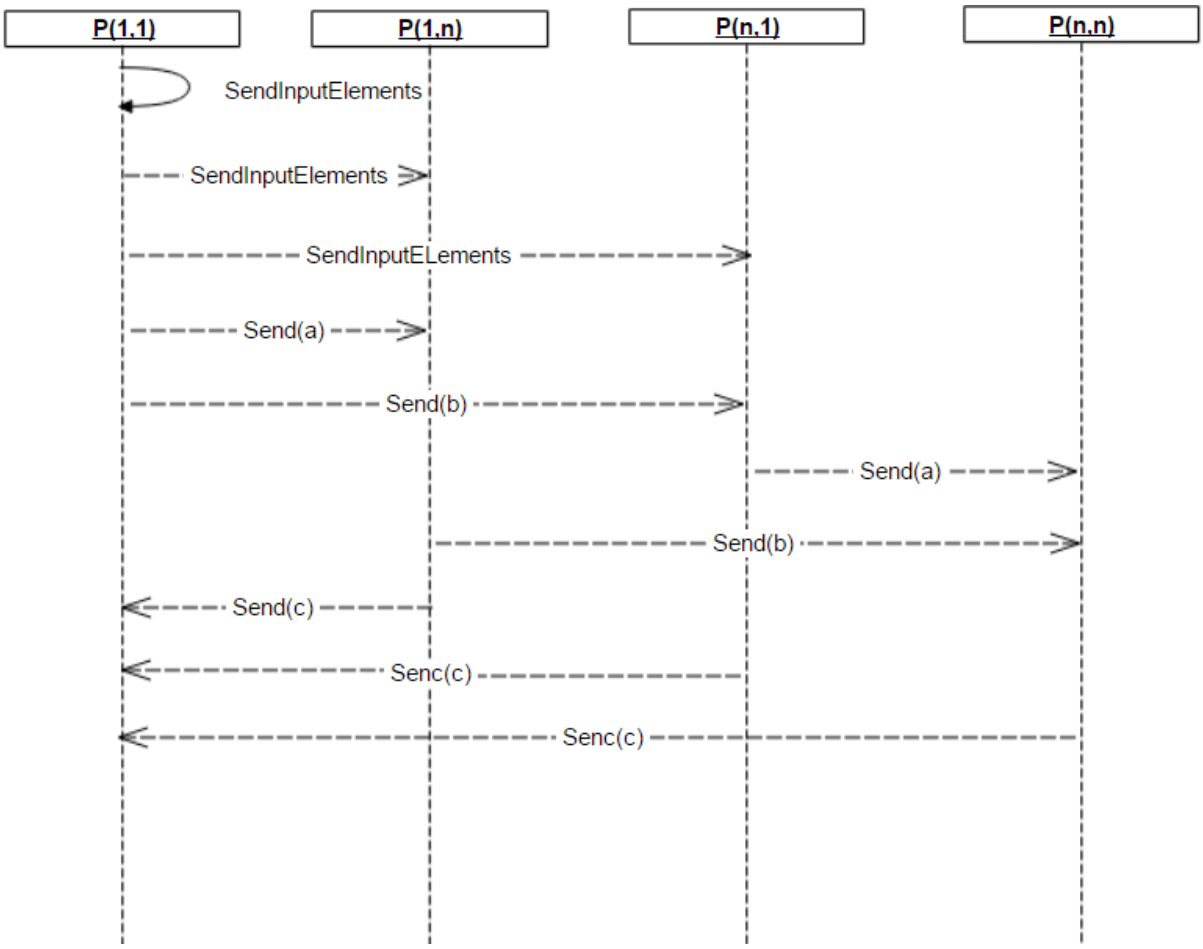


Obrázek 1: Doba běhu algoritmu v závislosti na počtu vstupních prvků

Závěr

Cílem projektu bylo analyzovat a implementovat paralelní algoritmus Mesh Multiplication a ověřit jeho časovou složitost. Testování potvrdilo, že čas potřebný k vynásobení matic roste s počtem prvků lineárně. To je patrné z grafu (Obr. 1). Komunikační protokol je přiložen v příloze.

Přílohy



Příloha 1: Sekvenční diagram znázorňující komunikační protokol