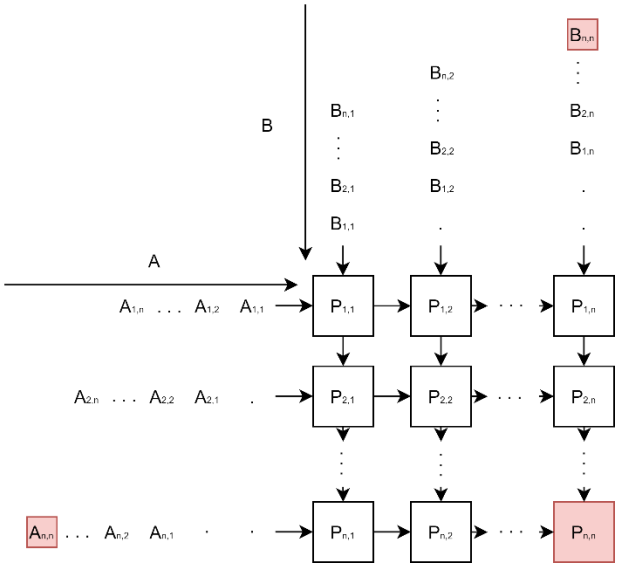
Mesh multiplication

PRL projekt 2
Havlíček Lukáš (xhavli46)

Algoritmus

Algoritmus pracuje s topologií mřížka o rozměrech KxM, kde K je počet řádků 1. matice a M je počet sloupců 2. matice. Při násobení matic AxB je matice A postupně posílána z boční strany mřížky a matice B postupně z vrchní strany mřížky. Každý procesor poté postupně přijímá prvky matice A od levého souseda a matice B od vrchního souseda, při přijetí těchto 2 prvků je mezi sebou vynásobí a přičte do vnitřní proměnné. Ve chvíli, kdy procesor zpracuje všechny přijímané prvky, tak v jeho vnitřní proměnné je výsledná hodnota matice pro prvek s indexem procesoru. Algoritmus je znázorněn na následujícím obrázku.



Složitost algoritmu

Vzhledem k tomu, že se u složitosti výpočtů s maticemi N udává jako délka jedné strany matice, bude odvození složitosti algoritmu předvedeno na násobení 2 matic, kde obě matice mají rozměr NxN. Algoritmus končí ve chvíli, kdy se k poslednímu procesoru (spodní pravá strana), dostanou poslední prvky z matice A (spodní levá strana) a matice B (vrchní pravá strana) viz vyznačené prvky a procesor v obrázku výše. K tomuto stavu se algoritmus dostane následovně.

První část lze uvážit, kdy se k nejspodnějšímu procesoru na levé straně mřížky dostane 1. prvek. K 1. procesoru se dostane první prvek v 1. kroku, ke 2. procesoru se dostane první prvek ve 2. kroku, takto lze postupně pokračovat až se k N. procesoru se dostane první prvek v **N. kroku**.

Druhou částí lze uvážit, kdy se tento prvek dostane k poslednímu procesoru (index N,N). V dalším (označíme jej znovu jako 1. krok této části) kroku se prvek posune z 1. na 2. procesor. V 2. kroku se prvek dostane na 3. procesor, takto můžeme postupovat až se tento prvek dostane na poslední N.

procesor v **N-1 krocích**. V tuto chvíli je tedy na procesoru$_{N,N}$ prvek $A_{1,N}$ a na procesoru$_{N,1}$ prvek $A_{N,N}$, tedy poslední prvek se právě dostal na první procesor ve své řadě.

Poslední třetí částí uvážíme, kdy se tento poslední prvek dostane na poslední procesor, což znamená ukončení algoritmu. V 1. kroku této části se prvek dostane na 2. procesor. V 2. kroku se dostane tento prvek na 3. procesor, takto můžeme pokračovat až se tento prvek dostane na poslední procesor v **N-1 krocích**.

Tyto 3 části následují po sobě, pro získání časové složitosti je potřeba je tedy sečíst: N + N-1 + N-1 = 3N – 1, což je O(n).

Počet procesorů je dán algoritmem, kde je vyžadována mřížka o rozměrech výsledné matice, což je v tomto případě NxN, což je $O(n^2)$.

Časová složitost: t(n) = O(n + n-1 + n-1) = O(n)
Počet procesorů: p(n) = $O(n^2)$
Cena: c(n) = t(n) * p(n) = $c(n * n^2) = c(n^3)$ (cena není optimální)

Implementace

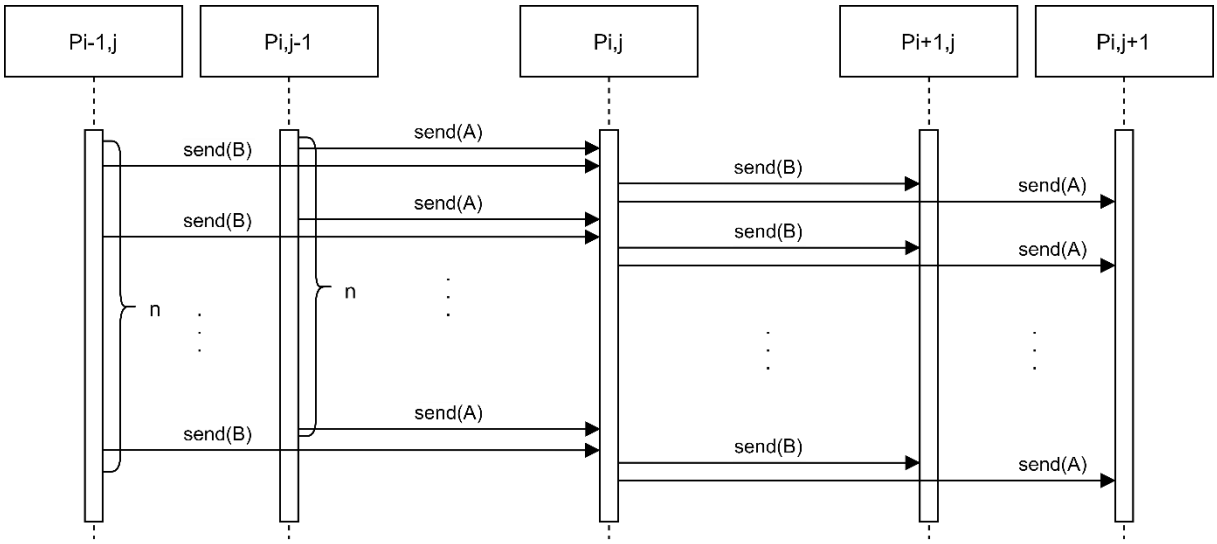
Program načítá 2 matice ze souborů „mat1“ a „mat2“, kde na prvním řádku souboru je uveden počet řádků matice1/počet sloupců matice2, toto načítání je řešeno hlavním procesorem (s id 0). Matice jsou zkontrolovány, zda mají na každém řádku, stejný počet hodnot a také zda počet sloupců 1. matice se rovná počtu řádků 2. matice (podmínka, která musí být splněna pro násobená matic). Následně tento procesor zašle všechny hodnoty z načtených matic příslušným procesorům.

Samotný výpočet je prováděn N procesy, kde každý procesor zná své ID a pomocí broadcastu od hlavního procesoru získá počet sloupců 1. matice (určuje počet očekávaných zpráv, které přijme) a počet řádků a sloupců výsledné matice, pro výpočet I a J procesoru. Každý procesor uvnitř cyklu přijímá postupně 2 zprávy s tagy A a B, při přijetí obou těchto zpráv je mezi sebou vynásobí, přičte k vnitřní proměnné a čeká na další dvě tyto zprávy. Po ukončení for cyklu je ve vnitřní proměnné výsledek matice pro indexy I a J, tuto hodnotu zašle procesor hlavnímu procesoru.

Hlavní procesor po ukončení výpočtu přijme hodnoty od všech procesorů s výslednými hodnotami a vytiskne je ve správném formátu na výstup.

Komunikační protokol

Na následujícím diagramu lze vidět komunikaci procesorů, kde send(x), značí MPI_send a MPI_recv jednoho čísla mezi procesory. Kde x je A nebo B v závislosti, jestli je prvek z matice A nebo B. Důležité je zde odesílání prvků procesorem $P_{i,j}$, kde odeslání send(A) a send(B), probíhá zároveň. Celkově každý procesor přijme a odešle N dvojích zpráv prvků matice A a B.



Závěr

Pro experimenty bylo využito matic o velikostech 2xN a Nx2, aby vše běželo na 4 procesorech (bez oversubscribe, kde by docházelo ke zkreslení kvůli přepínání kontextu). Počet prvků v grafu je udáno součinem 2xN. Z grafu lze vidět, že se podobá lineární závislosti, která byla odvozena jako časová složitost tohoto algoritmu. Pro měření reálného času bylo využito std::chrono, které bylo umístěno po načtení matic ze souboru a před výpis výsledku na výstup, aby byl zaznamenán pouze reálný čas a ne čas strávený nad IO operacemi.