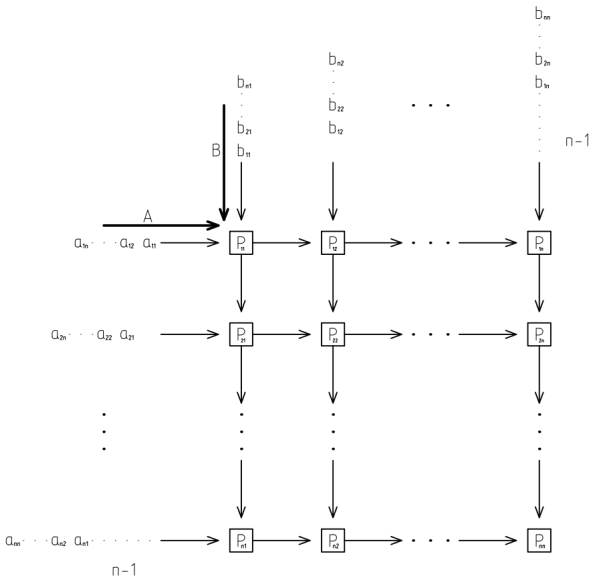
Paralení a distribuované algoritmy (PRL) 2020/2021
Mesh Multiplication

Daša Nosková (xnosko05)
xnosko05@stud.fit.vutbr.cz

30. apríl 2021

1 Rozbor algoritmu

Algoritmus **Mesh Multiplication** je paralelný algoritmus, používaný pre násobenie dvoch matíc (A a B) o rozmeroch $m \times n$ a $n \times k$. Algoritmus využíva $m \times k$ procesorov, ktoré sú uložené do tvaru mriežky. Do mriežky sa postupne privádzajú hodnoty z násobených matíc. Matica A je privádzaná zprava a matica B zhora. V mriežke sa nachádzajú súčiny matíc, teda každý procesor sa stará o jeden prvok z výslednej matice. Každý riadok/stĺpec i zaostáva o jeden prvok za riadkom/stĺpcom $i-1$ ako je vidieť na obr.1 Prvky sú spracovávané postupne posúvaním o jedno miesto v každom cykle.



Obrázek 1: Architektúra mriežky pre AxB

Na začiatku si každý procesor $P_{i,j}$, kde $i = \{1, ..., m\}$, $j = \{1, ...k\}$, vynuluje svoj register $C_{i,j}$ a následne každý procesor násobí dve hodnoty a a b, kým dostáva ďalšie hodnoty od matíc A a B. Súčin pričítáva k hodnote daného registru $C_{i,j}$, ktorý sa aktualizuje na novú hodnotu. Následne sú hodnoty matice A posielané procesoru $P_{i,j+1}$, hodnoty matice B procesoru $P_{i+1,j}$, až kým $i = m$ a $j = k$.

1.1 Analýza zložitosti

Prvky a_{m1} a b_{1k} sa dostanú k poslednému procesoru $P_{m,k}$ za $m+k-1$ krokov od začatia algoritmu, z čoho vyplýva lineárna časová zložitosť $O(n)$. Počet potrebných procesorov je $p = O(n^2)$, keďže počet procesorov je určený ako $m \times k$, pretože každému prvku výslednej matice prislúcha práve jeden procesor. Cena sa počíta ako $c(n) = t(n) \cdot p(n)$ a po dosadení

$$c(n) = (m+k-1) \cdot (mk) = m^2k + mk^2 - mk = O(n^3)$$

čo nie je optimálna cena.

2 Implementácia

Implementácia algoritmu je rozdelená do 3 krokov. Prvým krokom je načítanie matíc zo vstupných súborov a vypočítanie ich rozmerov pomocou funkcie `loadMatrix()`. Druhou časťou je rozposlanie všetkým procesorom rozmery načítaných matíc. Následne prvý procesor rozpošle časti matíc (vektorov), procesorom P_{1j} a P_{i1}, k čomu je potrebné vypočítať indexy daného procesoru cez vzorce 1 a 2, kde n = číslo procesoru.

$$i = n/k \tag{1}$$

$$j = n \mod k \tag{2}$$

Následne prvý procesor urobí z prvej matice m vektorov a z druhej k vektorov, ako je vidieť na obr. 1, ktoré po jednom rozpošle ďalším procesorom pomocou prechádzania celej matice vo vnorenom cykle, kde sa prechádza po riadkoch. Sebe samému proces prvky neposiela, rovno ich uloží do premennej. Index aktuálne posielaného prvku sa vypočíta ako

$$index = j + (i * c)$$

kde c = stĺpce aktuálne distribuovanej matice. Paralelne P_{1j} a P_{i1} príjmu dané vektory, ktoré uložia do premenných `row` (P_{i1}) a `col` pre (P_{1j}). Pomocou `MPI_Barrier(MPI_COMM_WORLD)` sa počká na všetky procesy a prechádza sa do samotného algoritmu, ktorý je implementovaný vo funkcii `multiply()`. Prvky sa spracovávajú kým nie je spracovaných n (zhodná strana matíc) prvkov. V prvom rade sa zas musí vypočítať i a j index procesoru. Procesory s P_{1j} dostávajú b hodnotu z dopredu naplneného vektora, inak je b hodnota prijatá od procesoru s číslom $j+(i-1)*k$ a s tagom `NEXT_ROW`. P_{i1} už majú prvky a a zvyšné procesory príjmajú hodnotu od procesoru $j-1+(i*k)$ s tagom `NEXT_COL`. Potom čo procesor obdrží obidve hodnoty, vynásobí ich, pričíta k premennej `mul`. Ak procesor nie je posledný v riadku alebo v stĺpci, pošle ďalšiemu procesoru. Hodnotu a pošle procesoru vypočítanému podľa vzorca 1 s tagom `NEXT_COL` a hodnotu b procesoru podľa 2 s tagom `NEXT_ROW`.

$$P = j + 1 + i * k \tag{1}$$

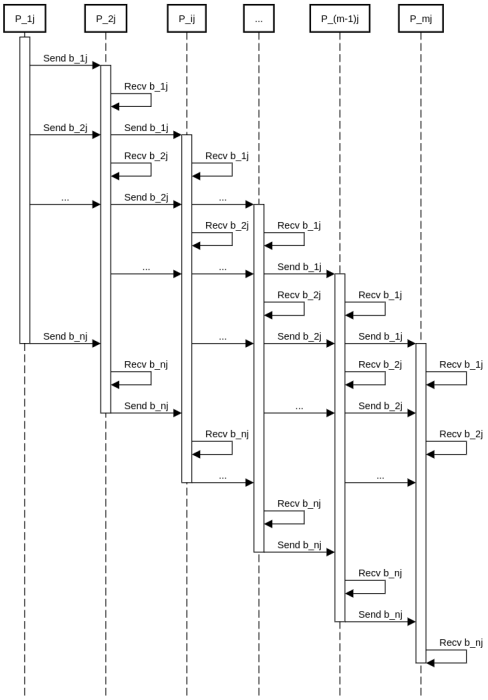
$$P = j + (i + 1) * k \tag{2}$$

Po násobení sa opäť cez `MPI_Barrier(MPI_COMM_WORLD)` čaká na všetky procesory. Nakoniec všetky procesory pošlú svoju výslednú hodnotu prvému procesoru, ktorý výsledok vypíše na výstup.

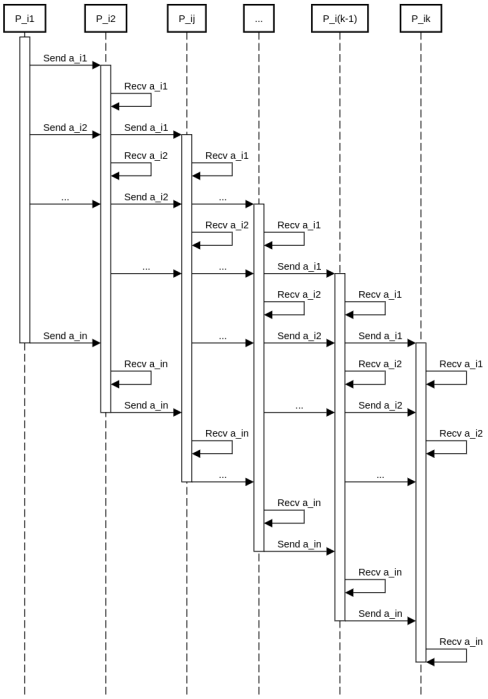
4 Experimenty

Boli vykonané experimenty s účelom overenia teoretickej časovej zložitosti trvania algoritmu v reálnom čase pomocou funkcie `chrono::high_resolution_clock::now()`. Matice boli rozdelené do dvoch skupín na "obdĺžnikové" a štvorcové. Prvá skupina bola testovaná pre hodnoty $i^2, i = \{2, 3, 4, 5\}$ a druhá pre hodnoty $2 * i, i = \{2, ..., 10\}$. Hodnoty v maticiach boli zvolené náhodne pomocou python funkcie `np.random.randint()` v rozsahu $(-20, 20)$ a pre obdĺžnikové matice bolo zvolené $n = 3$ a $m = 2$, k už podľa želaného počtu procesorov. Pre každý počet procesorov n bolo vykonaných 100 opakovaní, z toho 5% minimálnych a maximálnych hodnôt bolo odstránených. Zvyšné hodnoty sú vo výsledných grafoch priemerované. Ako je vidieť na obrázku 2 a obrázku 3 , reálna časová zložitosť nerastie úplne lineárne, ale napr. na obrázku 2 je vidieť medzi procesmi 10-18 lineárny rast.

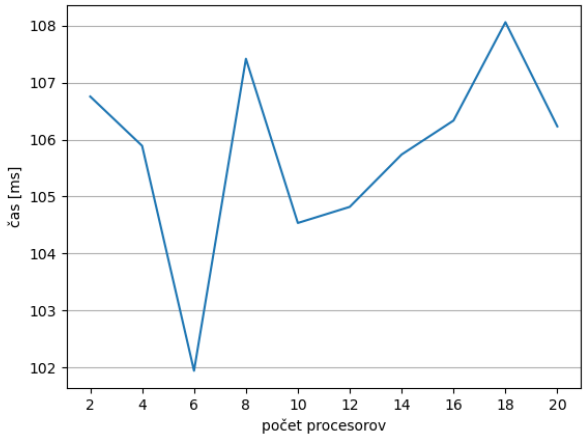
3 Komunikačný protokol



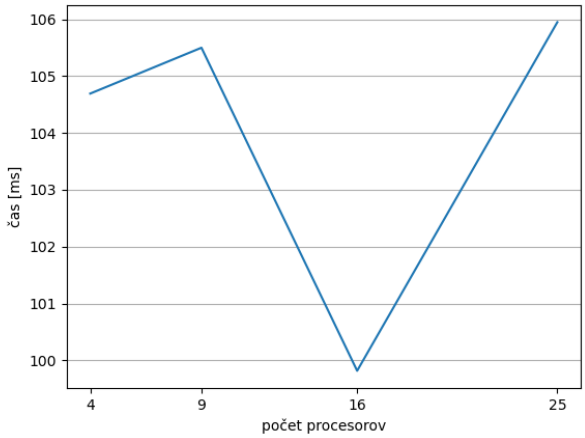
Obrázek 2: Komunikácia medzi procesmy v riadku



Obrázek 3: Komunikácia medzi procesmy v stĺpci



Obrázek 4: Graf pre obdĺžnikové matice



Obrázek 5: Graf pre štvorcové matice

5 Záver

V rámci projektu bol implementovaný algoritmus *mesh multiplication* a vykonané experimenty pre overenie jeho teoretickej časovej zložitosti. Experimenty neodpovedajú teoretickej časovej zložitosti, čo pripisujem použitiu malému počtu procesorov pri experimentoch.