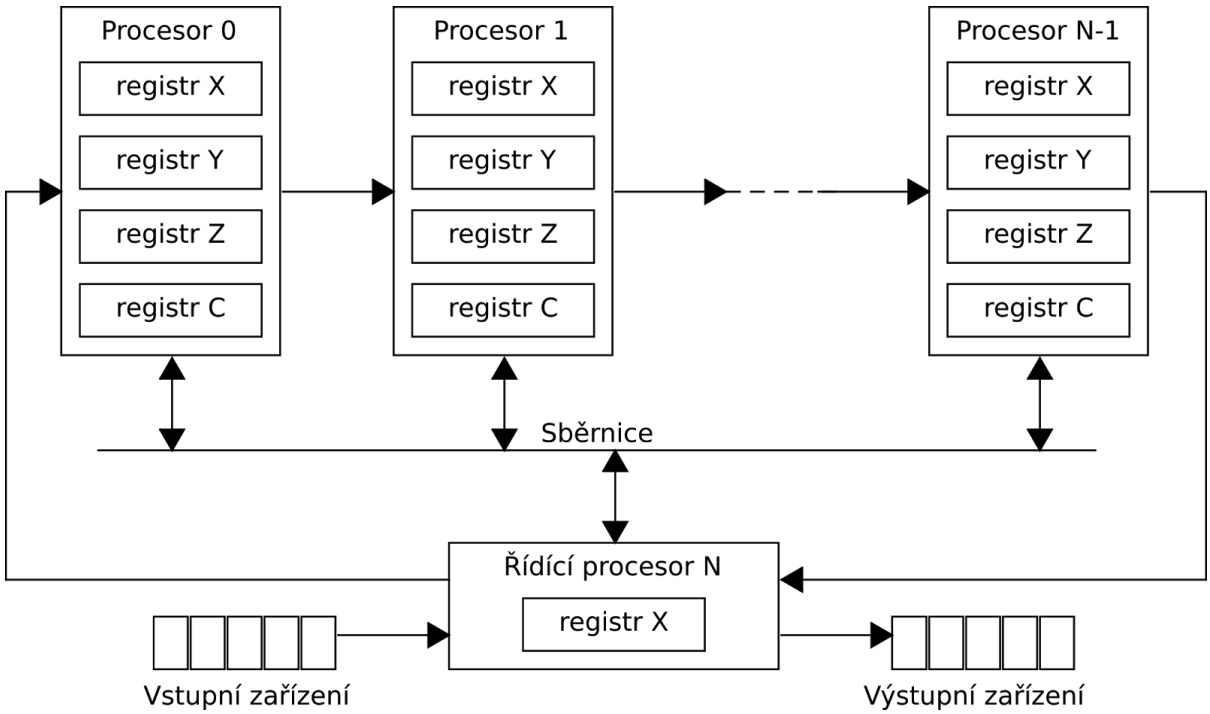
Rozbor a analýza algoritmu

Topologie

Enumeration Sort je paralelní řídící algoritmus, který při použití lineární topologii vyžaduje N procesorů, které jsou propojené lineárně mezi sousedními procesory a pomocí společné sběrnice. Tato sběrnice je schopná v každém kroku přenášet jednu hodnotu. Každý z procesorů disponuje 4 registry:

- **Registr X**, pro porovnávanou hodnotu
- **Registr Y**, pro všechny vstupní hodnoty, vůči kterým je **X** postupne porovnáno
- **Registr Z**, pro uchování výsledné hodnoty
- **Registr C**, pro určení a uchování relativního pořadí hodnoty **X** vůči řazené posloupnosti

Toto platí pro všechy procesory použité k řazení (procesory 0 až $N-1$). V našem případě je nutné neopomenout i procesor N (tedy "N plus první" procesor). Tento se v naší topologii chová jako řídící a jeho úkolem je distribuovat vstupní hodnoty a shromažďovat výsledky. Vystačí si s lineárním propojením na procesor 0 a $N-1$ a přístupem ke sběrnici. Řídící procesor dále disponuje 1 registrem na uchování čtených a shromážděných hodnot a přístupem ke vstupnímu a výstupnímu zařízení.Následující obrázek znázorňuje zvolenou topologii:



Princip algoritmu

Popisovaný algoritmus z přednášek[1] byl upraven, aby více vyhovoval topologii s řídícím procesorem.

1. Nastav pro každý procesor 0 až $N-1$ (dále prorovnávající procesor) hodnotu registru $C = 0$
2. Opakuj N krát, pro $0 \leq i < N$:
    a. Řídící procesor načte hodnotu ze vstupního zařízení, vypíše ji a následně pošle:
        - Procesoru 0 jako hodnotu Y

[1] https://www.fit.vutbr.cz/study/courses/PDA/private/www/h003.pdf , slide 24

- Procesoru i jako hodnotu X

b. Porovnávanící procesor 0 přijímá hodnotu Y od řídícího procesoru, ostatní tuto hondotu přijmou od předchozího v lineárním propojení
c. Pokud je $i = 0$ (během první iterace) každý porovnávací procesor přijme X hodnotu od řídícího procesoru
d. Pokud je u porovnávacího procesoru hodnota X i hodnota Y nastavena, porovnej je a inkementuj hodnotu registru C pokud:
   - $X < Y$, nebo
   - $X = Y$, tak zjisiti, zda-li $ID < i$, kde ID je pořadí procesoru v topologii
e. U každého porovnávacího procesoru pošli lineárním propojením hodnotu registru Y svému sousedovi

3. Opakuj N krát, pro $0 \leq i < N$:
   a. Synchronizuj procesy
   b. Pokud je $ID = i$, kde ID je pořadí procesoru v topologii (tedy procesor je na řadě), zjisti je-li $ID = c$. Pokud ano, přiřaď hodnotu registru X do výsledného registru Z
   c. Pokud $ID = i$ a zároveň $ID \neq C$ pošli hodnotu C na všem a hodnotu X procesoru s $ID = C$ (simulace sběrnice)
   d. Každý porovnávací procesor přijme ze sběrnice hodnotu, kterou porovná se svým ID. Pokud je to ID tohoto procesoru, přijmi hodnotu X
4. Opakuj N krát, pro $0 \leq i < N$:
   a. Pokud jsi řídící procesor, přijmi lineárním propojením hodnotu a vypiš ji
   b. Pokud jsi porovnávací procesor, pošli svou hodnotu registru Z po lineárním propojení a přijmi od předchozího procesoru hodnotu, kterou si ulož jako hodnotu registru Z

Algoritmus navržený v předáškách prošel úpravami, co se týče implementační a topologické věrnosti. Dále byly zapracovány specifika použité knihovny OpenMPI, které například umožňuje simulovat sběrnici jako *broadcast* vysílání identifikátoru, na který je hodnota směřována a její následné poslání normálním jednosměrným kanálem.
Dále algoritmus upravuje chování pro více stejných hodnot přečtených ze vstupního zařízení. Původní algoritmus tuto situaci neřeší vůbec. Navržená změna přidává porovnání, kdy v případě že $X = Y$ se provede následně porovnání $ID < i$. Tím je docíleno, že při porovnávání stejných hodnot je hodnota, která byla ve vstupu dříve, uložena směrována na procesor s vyšším ID, čímž je také dříve na výstupu. Díky této změně je algoritmus stabilním řazením.

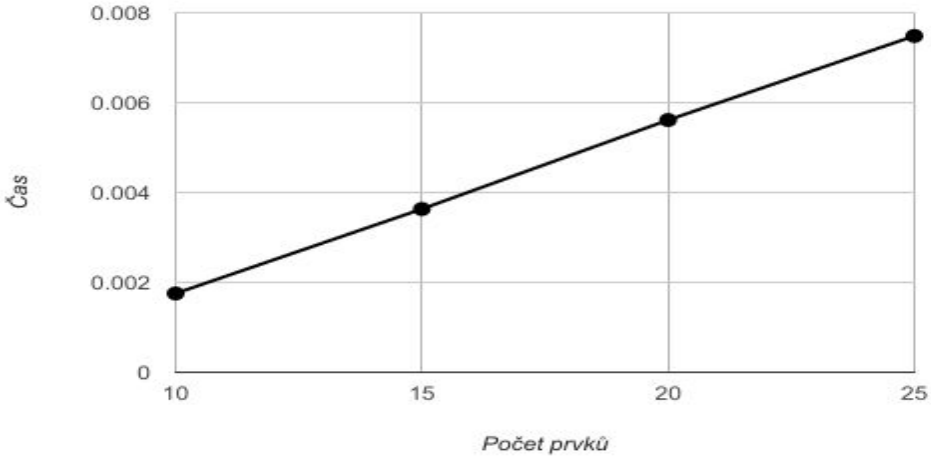
Složitost algoritmu

Asymptotická složitost algoritmu[2] je:	$O(1) + O(n) + O(n) + O(n) = O(n)$
Časová složitost	$O(n)$
Prostorová složitost	$O(n)$
Celková cena	$O(n) * O(n) = O(n^2)$

Pro experimentální ověření těchto hodnot proběhlo několik testů. Pomocí funkce knihonvy OpenMPI, funkce MPI_Wtime(), byla změřena doba trvání běhu programu pro různé počty hodnot. Toto měření bylo opakováno 100x pokaždé s jinými hodnotami a byla uvažována průměrná hodnota. Měřen byl průběh celého programu, přičemž režie spojená se zavedením knihovny OpenMPI a manipulace s vstupním souborem byla zanedbána a považována jako konstantní dobu trvající operace.

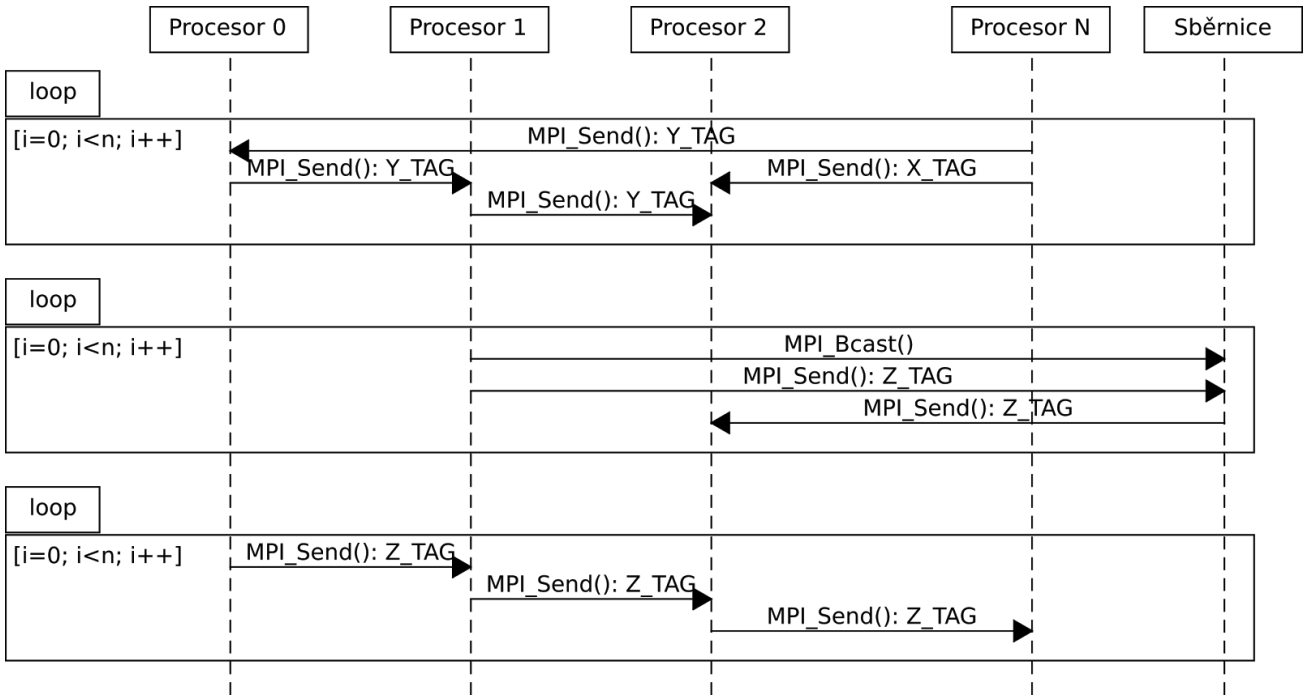
Počet prvků	Průměrný čas výpočtu (ze 100 opakování)
10	0.001753
15	0.003632
20	0.005612
25	0.007483

[2] Každá složka součtu odpovídá jednomu kroku podle uvedeného algoritmu



Komunikační protokol

Pro komunikaci mezi procesory disponuje knihovna OpenMPI funkcemi MPI_Send() an MPI_Recv(). Pro boradcast je k dispozici MPI_Bcast().



Závěr

Experimentální výsledky odhadované složitosti hypotézu potvrzují. Použitá knihovna OpenMPI pro komunikaci se osvědčila a samotná komunikace je znázorněna na diagramu v předchozí sekci.