Rozbor a analýza algoritmu

Pipeline merge sort je paralelní řadící algoritmus, který pro vstup o velikosti n vyžaduje $\log_2 n + 1$ lineárně propojených procesorů, ve kterém lze funkčně odlišit první procesor, poslední procesor a zbytek procesorů. Prostorová složitost algoritmu je tedy logaritmická $O(\log n)$. Principem algoritmu je slučování dvojic uspořádaných posloupností, jejichž délka je dána pořadím procesoru.

- První procesor P_0 načítá neseřazená data ze vstupní linky a data dále posílá pomocí dvou linek střídavě dalšímu procesoru. Po n krocích tak odešle všech n prvků.
- Následující procesory $\{P_i | i \in \langle 1; \log_2 n - 1 \rangle\}$ přijímají částečně seřazené sekvence o délce 2^{i-1} pomocí dvou linek, přičemž začít porovnávat data, spojovat sekvence a odesílat data dál mohou po obdržení prvního prvku z druhé seřazené sekvence. Ten se k procesoru dostane po $2^i + i - 1$ krocích. Data posílají dalšímu procesoru opět pomocí dvou linek.
- Poslední procesor přijímá taktéž data pomocí dvou linek, avšak na výstupu již má pouze jednu linku, kam umisťuje jedinou konečnou seřazenou sekvenci. Jelikož předchozí procesor P_i obdržel data, po kterých mohl začínat odesílat své seřazené sekvence v $2^i + i - 1$ krocích, může začít odesílat v dalším kroku, na poslední procesor se tak dostanou všechna data po $2^i + i + n - 1$ krocích, kde $i = \log_2 n$. Lze upravit na $n + \log_2 n + n - 1 = 2n + \log_2 n - 1$, časová složitost je tedy lineární $O(n)$.

Jelikož je časová složitost lineární a prostorová logaritmická, je tak celková cena algoritmu $O(n \times \log n)$, což je optimální cena.

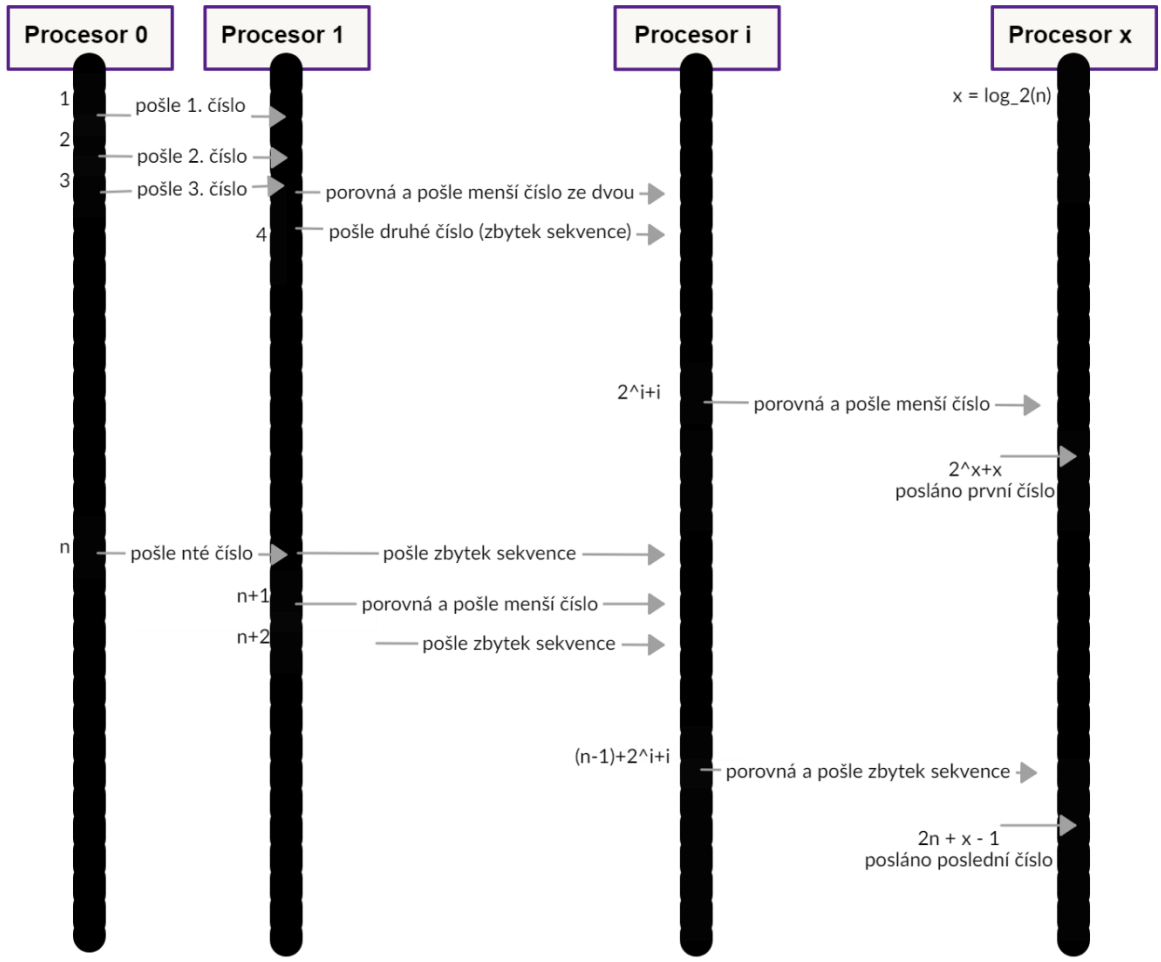
Implementace

Pro implementaci byl použit jazyk C/C++ s knihovnou Open MPI dle zadání. Každý procesor volá jednu ze tří funkcí dle toho, zda se jedná o první, poslední, nebo některý z prostředních procesorů.

- První procesor načítá data ze souboru *numbers*, která neuspořádaně vypisuje a následně je posílá dalšímu procesoru. Jelikož musí být nejprve odeslány všechny prvky před tím, než se může dostat první prvek na konec, nehrozí, že by výpis neproběhl ve správném pořadí.
- Prostřední procesory využívají dvou front. Nejprve naplní jednu frontu první sekvencí, kterou zašle předchozí procesor. Následně střídavě ukládá další sekvenci do druhé fronty, další sekvenci opět do té první a tak dále, po každém načteném prvku posílá menší z dvou front dalšímu procesoru – až do sloučení a odeslání obou sekvencí do jedné dalšímu procesoru. Po opakování do chvíle, kdy bylo přijato všech (dle zadání 16) prvků od předchozího procesoru jsou nadále již pouze kontrolovány dvojice hodnot z obou front a odesílány dalšímu procesoru. Po vyprázdnění obou front procesor končí svou práci.
- Obdobně jako předcházející procesory využívá tento procesor dvou front. Nejprve čeká na příjem dat od předchozího procesoru. Poté co začnou data přicházet přijme polovinu z celkového počtu (tedy zde 8) prvků a uloží je do fronty 1. Následně ukládá druhou polovinu prvků o druhé fronty, u každého prvku porovná vrcholky front a vypíše menší z nich. Po přijetí všech prvků porovnává vrcholky front a vypisuje menší z nich až do jejich vyprázdnění, čímž úspěšně seřadí obě posloupnosti do konečné jedné seřazené posloupnosti.

Komunikační protokol

Komunikace mezi procesory je graficky znázorněna na přiloženém sekvenčním diagramu.



Závěr

Při testování funkčnosti implementace bylo vyzkoušeno několik variant sekvencí, kromě náhodného řazení tak také již seřazené sekvence nebo opačně seřazené sekvence. Dle předpokladu však nemá předběžné řazení vliv na rychlost algoritmu, jelikož množství práce zůstává vždy velmi podobné, a tak jsou rozdíly v rámci statistických odchylek neměřitelné.