Pipeline Merge Sort

PRL projekt 1
Havlíček Lukáš (xhavli46)

Algoritmus

Pipeline Merge sort funguje na principu spojování dvou seřazených posloupností v lineární topologii s počtem procesorů log n +1. Každý procesor (kromě prvního) má dvě fronty o délce 2^{i-2}, kde i je číslo procesoru (od 1). První procesor má na vstupu pouze jednu frontu všech prvků a postupně je posílá následujícímu procesoru. Procesor přijímá od předchozího procesoru prvky a ukládá si je do první fronty, dokud ji nenaplní, v případě naplnění fronty (počet prvků je 2^{i-2}) je začne procesor ukládat na druhou frontu, dokud ji taktéž nezaplní a začne opět ukládat na první frontu. Procesor začne zároveň odesílat, když má na jednom vstupu „plnou" frontu a na druhém vstupu 1 prvek. Procesor odesílá větší(/menší) prvek zepředu svých front, zároveň si procesor musí pamatovat kolik prvků z dané fronty odeslal, aby odeslal z fronty maximálně 2^{i-2} prvků, aby neodeslal prvek, který mezitím přijmul, a patří do již nové posloupnosti.

Složitost algoritmu

Procesor P_i začne, když má na jednom vstupu 2^{i-2} prvků a na druhém 1, tedy začne $2^{i-2} + 1$ cyků po procesoru P_{i-1}. Procesor P_i tedy začne v celkovém cyklu $1 + \sum_{j=0}^{i-2} 2^j + 1 = 2^{i-1} + i - 1$ a skončí v cyklu $n - 1 + 2^{i-1} + i - 1$.
Pokud za i dosadíme poslední procesor, tj log n +1, tak vznikne $n - 1 + 2^{\log n+1-1} + \log n + 1 - 1$, po úpravě tedy $n - 1 + 2^{\log n} + \log n$, což se rovná 2n + log n – 1 počtu cyklů kdy skončí celý algoritmus.

Časová složitost: t(n) = O(2n + log n - 1) = O(n)
Počet procesorů: p(n) = log n + 1
Cena: c(n) = t(n) * p(n) = c(n * (log n + 1)) = c(n*log n) (cena je optimální)

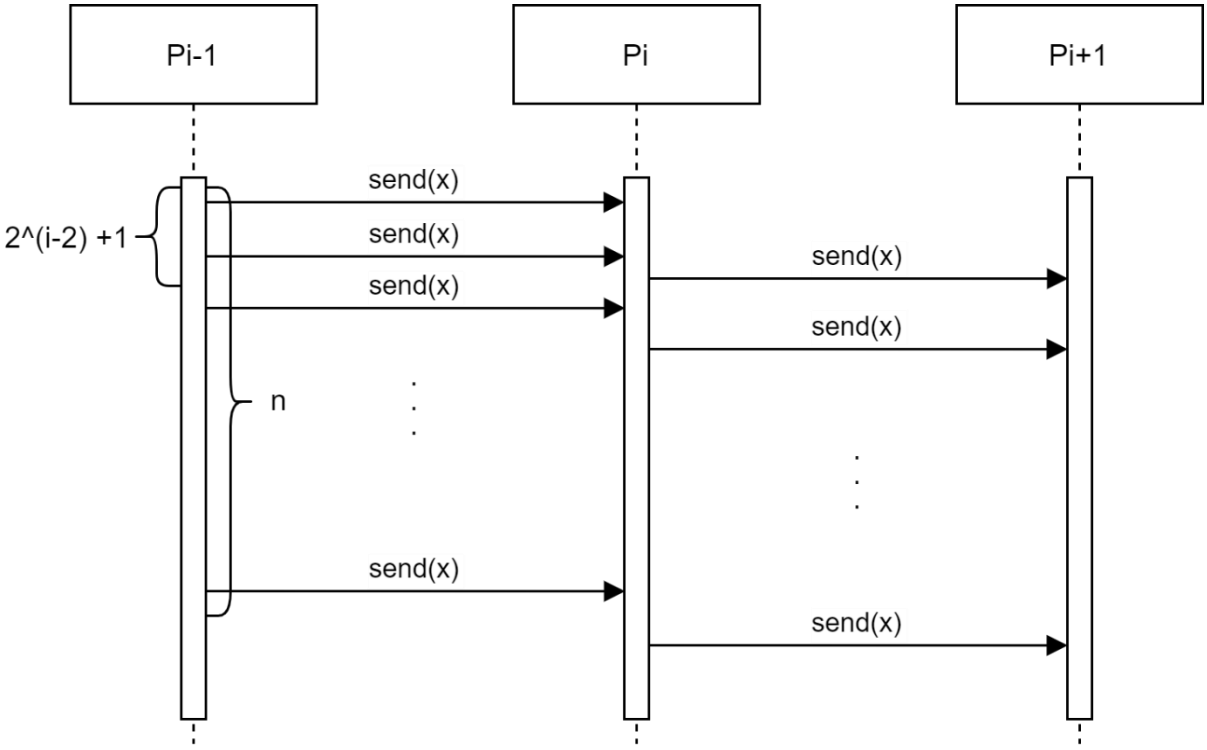
Implementace

Program načítá 16 čísel ze souboru „numbers" a pomocí 5 procesorů je seřadí, je implementován v C++ pomocí OpenMPI. Program je rozdělen na 2 hlavní části:
1. část – vykonává pouze jeden procesor (master), obsahuje načtení čísel ze souboru, kontrolu počtu čísel a procesorů. Poté tato čísla ze souboru posílá dalšímu procesoru v pořadí.
2. část – vykonávají všechny ostatní procesory, každý procesor zde má for přes n, pro přijetí všech čísel od předchozího procesoru. Uvnitř tohoto foru je také řešeno odesílání čísel dalšímu procesoru, pokud je jedna fronta „plná" a na druhé je alespoň 1 prvek. Pokud je tato podmínka splněna, tak procesor porovná prvky na začátku fronty a odešle dále menší z nich a z pomocné proměnné (nastavené na max. délku fronty) si odečte 1, aby z fronty neodeslal prvek, který patří již do nové posloupnosti a také pro kontrolu kdy jsou všechny prvky z dané posloupnosti odeslány (obě pomocné

proměnné front se rovnají 0). Ale protože odesílání prvků začíná později než příjímání, tak na konci tohoto foru ještě nejsou všechna čísla odeslána, a proto je zde ještě dokončeno odesílání těchto zbylých čísel. Výjimkou je zde poslední procesor, který už přijatá čísla dále neposílá, ale místo zaslání je vypíše na výstup.

Komunikační protokol

Na následujícím diagramu lze vidět komunikaci procesorů, kde send(x), značí MPI_send a MPI_recv jednoho čísla mezi procesory. Je zde možno vidět, že procesor Pi začíná s odesíláním až po přijetí 2^{i-2} + 1 prvků a každý procesor celkem odešle a přijme n čísel.



Závěr

Algoritmus Pipeline Merge Sort je optimálním algoritmem (viz cena algoritmu) a na lineární topologii nelze dosáhnout lepší časové složitosti než O(n), kterou má tento algoritmus. Při testování byl využit skript test.sh, který vygeneruje soubor s náhodnými 16 čísly. Pro testování reálného času řazení bylo využito clocku z std::chrono, umístěn za načtení hodnot ze souboru a ukončení řazení, ale protože se jedná pouze o 16 čísel, tak měření času nedává příliš smysl a při testování se hodnoty pohybovaly od 200 do 800 µs na stejném vstupu. Díky tomuto nebylo možné provést přesvědčivé experimenty na různých posloupnostech (seřazená, náhodná a seřazená naopak), ale k povaze algoritmu bude řazení těchto posloupností trvat stejnou dobu.