# PRL - Projekt 1. - Odd-Even Merge Sort

Michal Rein

4. dubna 2022

## 1 Rozbor a analýza algoritmu

Odd-Even Merge Sort je speciálním druhem algoritmu pro paralelní řazení posloupnosti čísel. Ke svému fungování vyžaduje specificky propojené výpočetní jednotky, kde každá jednotka má vždy 2 vstupy, 2 výstupy a schopnost porovnávat číselné hodnoty. Tyto jednotky jsou navzájem provázány a postupně kaskádově spojovány do větších síťových bloků, kdy každý blok třídí a spojuje seřazené sekvence lichých a sudých prvků. Délku vstupní sekvence je vhodné volit dle vztahu $n = 2^m$. Příklad sítě pro řazení 8 prvků lze vidět na obrázku 1.

### 1.1 Analýza algoritmu

Počet potřebných procesorů je dán sumou výpočetních jednotek v každé $N \times N$ síti. Obecně v závislosti na délce vstupní sekvence $n = 2^m$ lze síť rozdělit do fází, kde:

- 1. fáze (Blok 1x1 sítí) vyžaduje $2^{m-1}$ 1x1 sítí po 1 procesoru
- 2. fáze (Blok 2x2 sítí) vyžaduje $2^{m-2}$ 2x2 sítí po 3 procesorech
- 3. fáze (Blok 4x4 sítí) vyžaduje $2^{m-3}$ 4x4 sití po 9 procesorech
- 4. fáze (Blok 16x16 sítí) vyžaduje $2^{m-4}$ 16x16 sítí po 25 procesorech
- ...

Počet potřebných procesorů na zpracování vstupní sekvence o délce $n$ je tedy $p(n) = \mathcal{O}(n.log^2 n)$. Výsledná časová složitost algoritmu je dána vztahem $t(n) = \mathcal{O}(m^2) = \mathcal{O}(log^2 n)$, cena pak není optimální a je dána vztahem $c(n) = p(n).t(n) = \mathcal{O}(n.log^4 n)$.

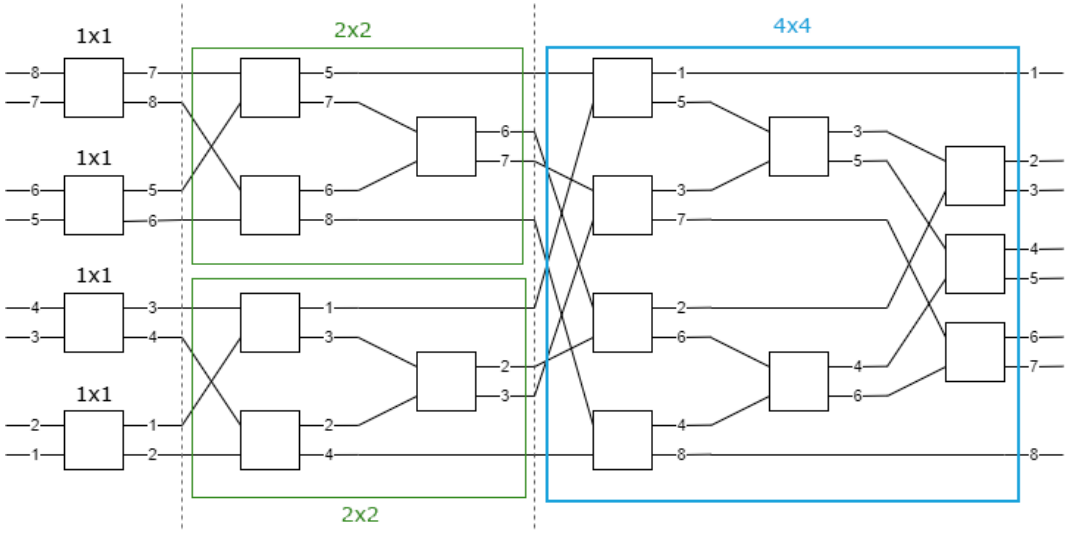
## 2 Implementace

Algoritmus je implementován v jazyce C s využitím knihovny *Open MPI* jako prostředek pro komunikaci mezi procesy. Zdrojový kód s implementací se nachází v souboru `oems.c`. Algoritmus je navržen pro řazení 8 prvků, které jsou načítány ze souboru `numbers` jako 1 bytové hodnoty typu *unsigned char*. K provedení řazení je zapotřebí celkem 19 procesů.

V první fázi programu dochází k rozdělení procesů dle svého umístění v bloku (vrstvy) $N \times N$ sítí. Každý z procesů dostane přiřazenou konkrétní hodnotu *layer*, na základě které je komunikátor světa (*MPI_COMM_WORLD*) rozdělen a dochází k vytvoření menších komunikátorů, ve kterém jsou vždy přítomny pouze procesy konkrétního bloku $N \times N$ sítě. Pro řazení 8 prvků jsou celkem vytvořeny 3 vrstvy (analogicky dle schématu sítě na obrázku 1).

Po jednoznačném odlišení procesů dochází ke čtení vstupního binárního souboru procesem *MASTER* (proces s označením 0), který následně pomocí metody *MPI_Scatter()* rozešle vždy 2 prvky procesům vstupního bloku (tohoto bloku je proces *MASTER* rovněž součástí). Procesy ostatních bloků mezitím čekají u bariéry na dokončení I/O operace.

Po dokončení distribuce hodnot do vstupního bloku, zahájí všechny procesy sekvenci popsanou metodou *recv_process_send()*. Každý proces volá tuto metodu s unikátním mapováním ranků pro příjem a odeslání hodnot. Celá sekvence probíhá pomocí asynchronních komunikace následovně:




Obrázek 1: Schéma výsledné sítě pro řazení 8 elementů.

1. Proces čeká na příjem 2 vstupních prvků
2. Proces zkontroluje, zda jsou oba prvky seřazeny, případě přehodí jejich pořadí
3. Proces odesílá seřazené prvky novým příjemcům

Procesy, které obdržely vstupní hodnoty prostřednictvím *MPI_Scatter()* od *MASTER* procesu provádí pouze 2. a 3. krok. Takto hodnoty postupným řazením a spojováním probublají do výsledné seřazené sekvence. Jednotlivé seřazené prvky jsou odeslány opět pomocí metody *recv_process_send()* procesu *MASTER*, který mezitím očekává přijetí zprávy a dat od koncových procesů. Po přijetí dat od všech koncových procesů je výsledek uložen do pole, jehož obsah je následně vypsán na *stdout*.

# 3 Závěr

Výsledná implementace algoritmu Odd-Even Merge Sort funguje dle očekávání.