1 Merge-splitting sort

Algoritmus Merge-splitting sort slouží k paralelnímu výpočtu v lineárním poli procesorů. Každý procesor se stará o více prvků, přesněji **n/p** prvků, kde **n** je počet všech prvků a **p** je počet procesorů. Je podobný algoritmu Odd-even transposition sort, ale porovnání a výměna je zde nahrazena operacemi merge-split.

1.1 Algoritmus

```
Algoritmus
for i = 1 to p do in parallel
        procesor P_i seřadí svou posloupnost sekvenčním algoritmem
endfor
for k = 1 to ⌈p/2⌉ do
        1) for i = 1, 3, ..., 2.⌊p/2⌋ do in parallel
                spoj S_i a S_i+1 do setříděné sekvence S_i'
                S_i = první polovina S_i'
                S_i+1 = druhá polovina S_i'
           endfor
        2) for = 2, 4, ... , 2.⌊p/2⌋ do in parallel
                ....
           endfor
endfor
```

Obrázek 1: Algoritmus Merge-splitting sort

1.2 Analýza složitosti

Dle obrázku 1 musí v prvním kroce všechny procesory provést seřazení svým prvků pomocí optimalizovaného sekvenčního algoritmu, složitost toho kroku je `O((n/p) * log(n/p))`. Přenosy hodnot do procesoru a zpět mají oba složitost `O(n/p)`. Spojení původní a přijaté hodnoty s pomocí optimalizovaného algoritmu pak `O(2 * n/p)`. Ve finále tedy má krok 1 nebo 2 složitost `O(n/p)`.

Výsledné složitosti jsou tedy:

- Časová složitost: `t(n) = O((n * log n)/p) + O(n)`
- Cena: `c(n) = O(n * log n) + O(n * p)`

Algoritmus je optimální pokud $p \leq \log n$.

2 Implementace

K implementaci se použil jazyk C++ a knihovna Open MPI.

2.1 Načtení hodnot

První (nultý) procesor načte ze souboru posloupnost čísel a rozešle je sobě i všem ostatním procesorům. Před tím se ještě ověří, zda je posloupnost beze zbytku dělitelná počtem procesoru (každý procesor se stará o stejný počet čísel), v případě, že není, tak se provádí korekce a k seznamu řazených čísel se přidávají 0, tak aby doplnily počet do dělitelnosti. Shodně to funguje i pokud je počet čísel menší než počet procesorů.

Na konci tohoto kroku se na standartní výstup vypíšou načtené hodnoty.

2.2 Sekvenční seřazení

Po tom, co procesory přijmou své hodnoty, tak je seřadí pomocí standartní funkce `sort`.

2.3 Posílání hodnot a merge-split

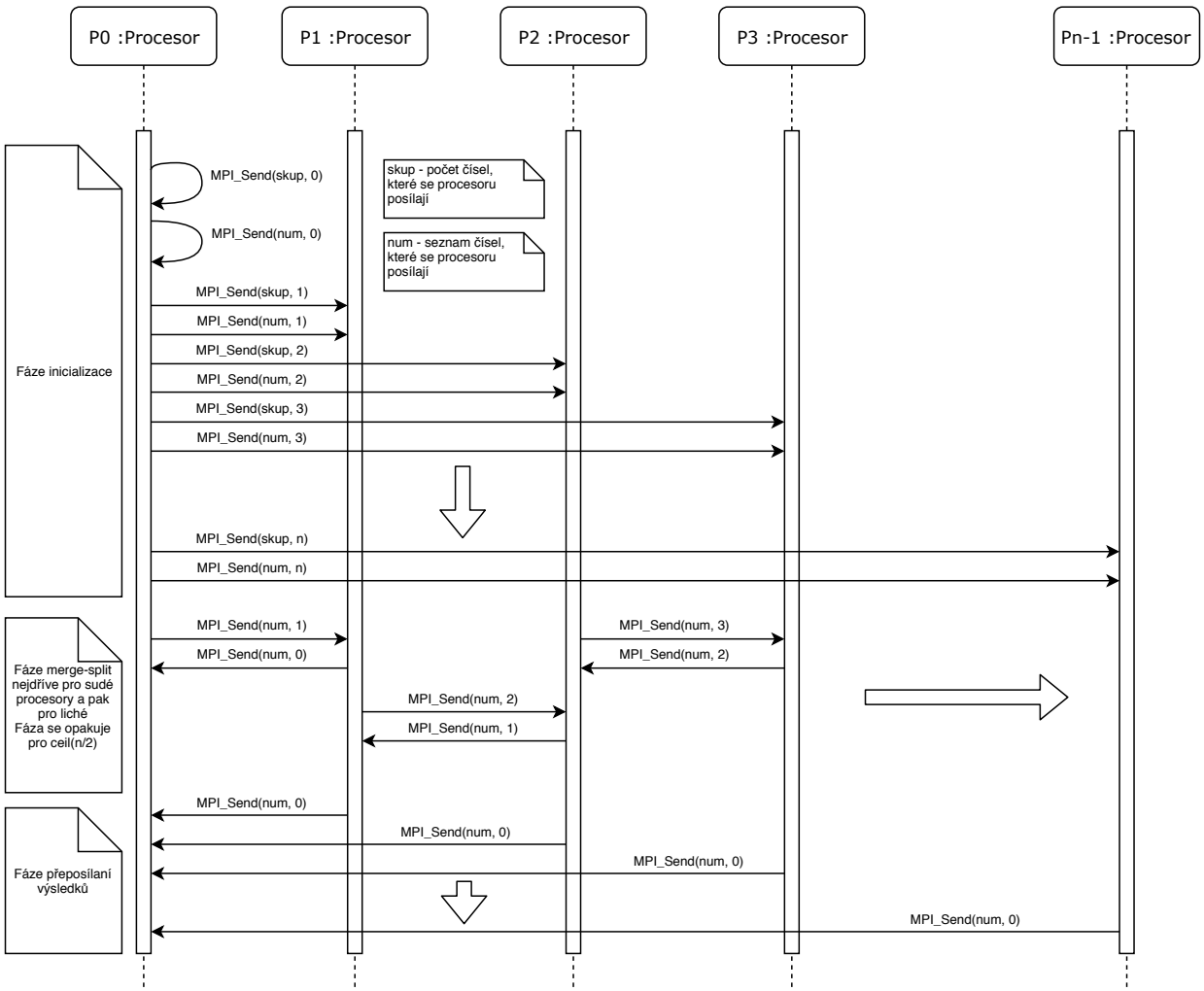
Následně procesory (nejdříve sudé a pak liché procesory) posílají své hodnoty do svého pravého souseda. Pravý soused po příjmu hodnot provede merge svých a přijatých hodnot pomocí standartní funkce `merge`. Výsledné pole hodnot se rozdělí na 2 poloviny a levá část se pošle (levému) sousedovi zpět. Pravá část se nastaví jako nové hodnoty procesoru (krok 1 a 2 algoritmu 1)

Tato část algoritmu se opakuje do $\lceil n/2 \rceil$ Tato hodnota je vypočítána pomocí standartní funkce `ceil`.

2.4 Výsledné hodnoty

Na konci všechny procesory (kromě prvního – nultého), pošlou nultému procesoru své hodnoty. Ten si je uloží, a nakonec je v požadované podobě vypíše na standartní výstup. Pokud vstupní posloupnost čísel nebyla beze zbytku dělitelná počtem procesorů, tak se skutečné seřazené hodnoty nacházejí až za pomocnou posloupností nul (pomocná posloupnost se nevypisuje).

2.5 Sekvenční diagram



Obrázek 2: Sekvenční diagram komunikačního protokolu

3 Experimenty se vstupy

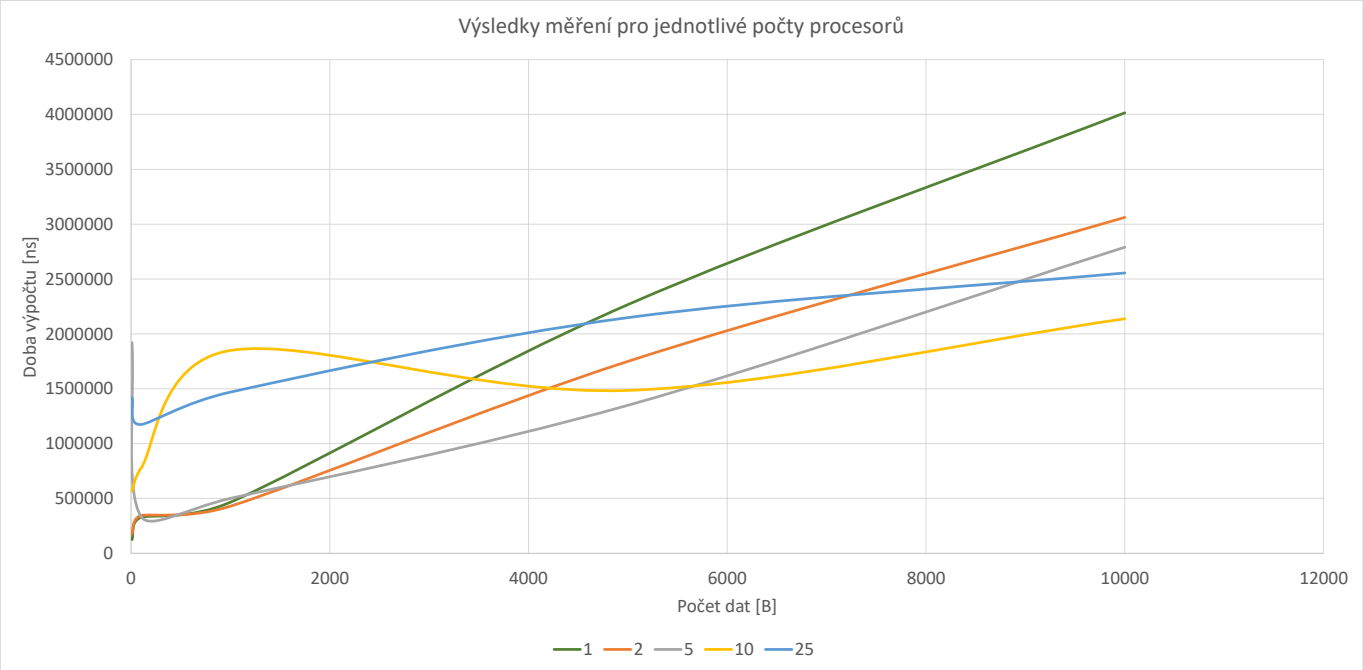
Experimentování a měření probíhalo na školním serveru Merlin. Merlin je víceuživatelský systém obsahující jeden procesor s 12 logickými procesory, výsledné časové hodnoty z experimentování tedy mohou být zkreslené z důvodu vytížení systému.

K měření času probíhání výpočtu se použila funkce `int clock_gettime(clockid_t clk_id, struct timespec *tp)`, algoritmus měření byl převzat z Profiling Code Using clock_gettime. Výpočet provádí vždy nultý procesor, první hodnotu nastaví po přečtení souboru s čísly a jejích vypsání, druhou po tom, co přijme všechny hodnoty od ostatních, ale ještě předtím, než je vypíše.

Měření probíhalo s různým počtem vzorků a různým počtem nastavených procesorů. Pro každou kombinaci se provedlo 7 měření, odebralo se nejlepší a nejhorší výsledek a ostatní se zprůměrovaly. Lze je nalézt v tabulce 1. Časové výsledky jsou v nanosekundách. Na obrázku 3 lze pak nalézt graf s průměrnými výsledky doby výpočtu pro jednotlivé procesory.

Počet vstupních dat [B]	Počet procesorů				
	1	2	5	10	25
10	125122	178653	1920414	566416	1417883
100	327415	343814	334893	775264	1174060
1000	465815	429463	500520	1848778	1469356
5000	2266858	1749455	1349154	1484955	2148229
10 000	4014895	3061577	2789877	2137730	2555291

Tabulka 1: Výsledky měření



Obrázek 3: Graf s průměrovanými výsledky měření

4 Závěr

Z obrázku 3 je zřejmé, že naměřené výsledky přímo neodpovídají teoretické časové složitosti algoritmu Merge-splitting sort. Může to být způsobeno nepřesnou implementací požadovaného algoritmu nebo, jak již bylo dříve zmíněno, vytížením serveru merlin.