PRL Projekt 2 – Enumeration sort

Vojtěch Havlena (xhavle03)

6. dubna 2017

1 Rozbor a analýza algoritmu

Algoritmus Enumeration sort využívá lineární spojení n procesorů, které je doplněno o společnou sběrnici. Vstupem je posloupnost čísel $(x_1, \ldots, x_n)$. Každý procesor i má k dispozici registry X_i, Y_i, Z_i a C_i. Samotný algoritmus lze potom popsat následujícími kroky:

1. Každý procesor i si nastaví hodnotu registru C_i na 1.
2. Následující kroky se v cyklu pro $1 \leq k \leq 2n$ opakují $2n$ krát:
   (a) V případě, že postupně zpracovávaná vstupní posloupnost ještě není vyčerpána (tj. $k \leq n$), prvek x_k se (sběrnicí) vloží do registru X_k, obsah registrů Y všech procesorů se posune doprava (lineárním spojením) a do registru Y_1 se (lineárním spojením) vloží x_k.
   (b) Každý procesor, který má již v registrech X a Y uložené hodnoty ze vstupu, provede porovnání těchto hodnot. Je-li $X > Y$, inkrementuje obsah registru C.
   (c) Po vyčerpání vstupní posloupnosti (tj. $k > n$) procesor P_{k-n} pošle obsah svého registru X procesoru $P_{C_{k-n}}$, který si tuto hodnotu uloží do svého registru Z.
3. V dalších n cyklech jednotlivé procesory postupně posouvají obsah svých registrů Z směrem doprava a procesor P_n produkuje výslednou seřazenou posloupnost.

Analýza složitosti

V prvním kroku algoritmu si paralelně všechny procesory nastaví hodnotu registru C na 1. Časová složitost tohoto kroku je tedy $\mathcal{O}(1)$. Kroky 2a, 2b a 2c provádí porovnání a poslání konstantního počtu hodnot. Tedy časová složitost těchto kroků je rovněž $\mathcal{O}(1)$. Provádění těchto kroků je opakováno celkem $2n$ krát. Celková časová složitost kroku 2 je tedy $\mathcal{O}(n)$. Posuv hodnoty v 3. kroku lze provést v konstantním čase a tento posuv se provádí v cyklu n krát. Časová složitost kroku 3 je tedy $\mathcal{O}(n)$. Celkovou časovou složitost algoritmu lze vyjádřit jako

$$t(n) = \mathcal{O}(1) + \mathcal{O}(n) + \mathcal{O}(n) = \mathcal{O}(n). \tag{1}$$

Vzhledem k tomu, že algoritmus pracuje na lineárním poli n procesorů, počet procesorů je $p(n) = n$. Celková cena je tedy

$$c(n) = p(n) \cdot t(n) = \mathcal{O}(n^2). \tag{2}$$

2 Implementace

Algoritmus byl implementován v jazyce C++ s využitím knihovny Open MPI. Implementace pro seřazení n hodnot využívá $n+1$ procesorů. Procesory jsou označeny jednoznačnými číselnými identifikátory z $\{0, \ldots, n\}$ (Rank). Procesory jsou pomocí svých hodnot Rank označeny jako $P_0, \ldots, P_n$ Procesor s P_0 (master) se na samotném řazení nepodílí, pouze posílá načtené hodnoty jednotlivým procesorům a na konci získává seřazenou posloupnost.

Pro správné odlišení zpráv je využíváno značení zpráv MPI pomocí položky TAG. Zprávy, které posílá master procesoru P_i pro uložení vstupní hodnoty x_i do registru X_i, jsou označeny `TAG_BUS`. Hlavní cyklus posouvání prvků vstupní posloupnosti a jejich porovnávání s aktuální hodnotou X probíhá v n krocích. Postupné posouvání vstupních hodnot mezi sousedními procesory (P_i a P_{i+1}) je umožněno zprávami, které jsou označeny `TAG_INPUT`. Po porovnání všemi vstupními hodnotami, procesor P_i pošle obsah X_i procesoru P_{C_i} pomocí zprávy označené jako `TAG_RESULT_INTER`. Nakonec je obsah registrů Z posouván pomocí zpráv označených `TAG_RESULT_OUT`.

Algoritmus byl rovněž upraven, aby byl schopen řadit posloupnosti obsahující stejná čísla. Hodnota C je inkrementována pokud $X > Y$ nebo v případě, že $X \geq Y$, ale to jen pokud pořadí prvku vstupní posloupnosti, který je uložen v Y, je menší než Rank procesoru.

3 Komunikační protokol

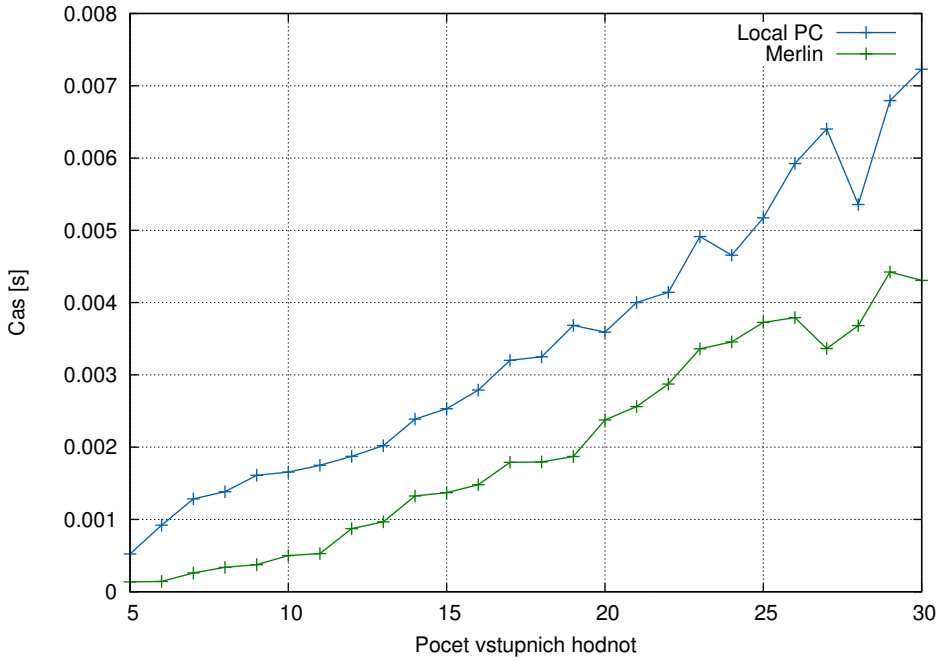
Jak již bylo popsáno v sekci Implementace, pro rozlišení zpráv se používá jejich značení (TAG). Komu se má zaslat zpráva záleží na hodnotě Rank procesoru. Na začátku algoritmu, procesor P_0 (master) posílá jednotlivé vstupní hodnoty x_k procesorům P_1 a P_k, kde $1 \leq k \leq n$. Dále každý procesor P_i, kde $1 \leq i \leq n-1$ postupně posílá hodnotu Y svému pravému sousedovi (procesoru P_{i+1}). Až procesor P_i provede porovnání se všemi vstupními hodnotami, posílá zprávu procesoru P_{C_i}. Poté, co všechny procesy (kromě P_0) obdrží hodnotu, každý procesor P_i postupně posílá hodnoty Z_i procesoru P_{i+1} (P_n posílá hodnoty procesoru P_0). Komunikační protokol je vizualizován na obrázku 2 (pro přehlednost je umístěn na poslední straně).

4 Experimenty

Cílem experimentů je ověřit, zda teoreticky odvozená časová složitost odpovídá reálné časové složitosti (době provádění algoritmu). Doba provádění algoritmu je měřena pomocí funkce `MPI_Wtime`. Do doby provádění algoritmu není započítáno

načítání vstupní posloupnosti a výpis seřazené posloupnosti na výstup. Doba provádění se začíná měřit až v okamžiku, kdy jsou všechny procesory připraveny zahájit řazení (toho je docíleno pomocí bariéry). Měření končí, když procesor s P_0 (master) obdrží seřazenou posloupnost.

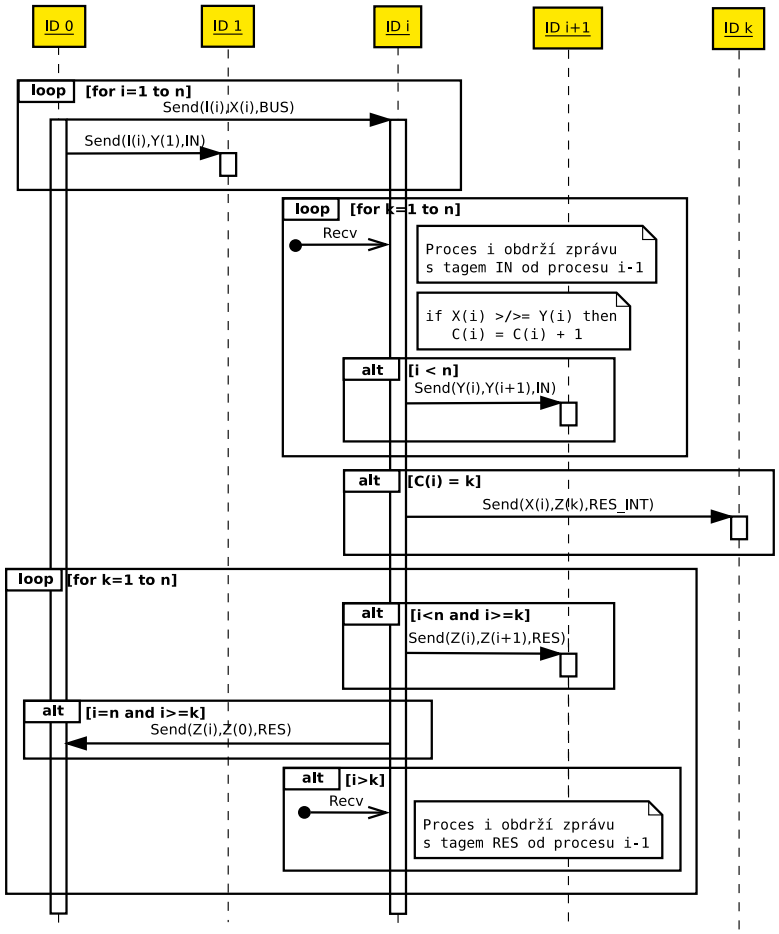
Experimenty byly provedeny pro velikosti vstupu $5-30$ hodnot. Měření probíhalo na lokálním stroji (Debian, Intel Core i5) a na serveru Merlin. Měření pro každou velikost vstupu bylo provedeno celkem 50 krát a tyto hodnoty byly zprůměrovány. Výsledky experimentů jsou uvedeny na obrázku 1.



Obrázek 1: Výsledky provedených experimentů.

5 Závěr

Cílem projektu bylo implementovat algoritmus Enumeration sort a ověřit, zda skutečná doba provádění odpovídá teoretické časové složitosti. Z experimentů vyplynulo, že pro menší počty vstupních hodnot (cca do 25) je průběh opravdu lineární, což odpovídá teoretické časové složitosti. Pro větší počet vstupních hodnot již dochází k odchýlení od lineárního průběhu. Navíc pro tyto vyšší hodnoty je i větší rozptyl naměřených časů. Odchylka může být způsobena režií posílání zpráv, přepínáním procesů nebo také aktuálním vytížením procesorů (časová složitost by měla být lineární v případě, že procesy poběží skutečně paralelně, což minimálně na lokálním stroji splněno nebylo).



Obrázek 2: Komunikační protokol pro n procesorů. Procesory jsou rozlišeny pomocí hodnoty Rank. Zasílání zpráv je označeno plnou šipkou s popisem Send($Odkud$, Kam, Tag), kde $Odkud$ je posílaná hodnota, Kam je místo uložení a Tag je zkratka označení zprávy (jednotlivá označení jsou popsána v sekci Implementace). Pomocí šipky s kulatým koncem s nápisem Recv je znázorněno přijetí zprávy (zpráva byla odeslána již dříve). Prvek na pozici i vstupní neseřazené posloupnosti je označen jako $I(i)$.