Paralelní a distribuované algoritmy – dokumentace k projektu 1

Vysoké učení technické v Brně

Petr Stehlík <xstehl14@stud.fit.vutbr.cz> 13. června 2018

1 Zadání

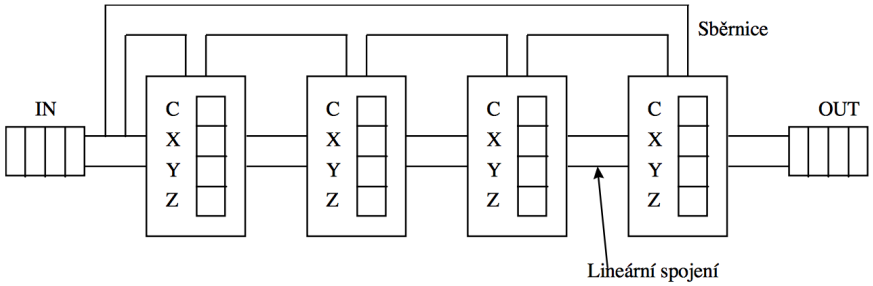
Cílem projektu byla implementace algoritmu enumeration sort na lineárním poli procesorů, který byl prezentován během přednášek. Běh a kompilace programu je zprostředkován pomocí skriptu `test.sh`. Implementace využívá knihovny Open MPI[2].

2 Rozbor a analýza algoritmu

Enumeration sort je algoritmus pro seřazení všech prvků $x_1 ... x_n$ pomocí nalezení konečné pozice všech prvků v poli určením počtu prvků menších než daný prvek x_i.

Enumeration sort na lineárním poli procesorů disponuje společnou sběrnicí pro všechny procesory. Tato sběrnice je schopna v každém kroku přenést jednu hodnotu a všechny procesory jsou doplněny lineárním spojením. Schéma zapojení procesorů je znázorněné na obrázku 1.

Každý procesor obsahuje celkem 4 registry: C – počet prvků menších než x_i, X – prvek x_i, Y – postupně prvky $x_1 ... x_n$, Z – seřazený prvek X_i.



Obrázek 1: Schéma zapojení procesorů. Převzato z [1]

Algoritmus byl modifikován tak, aby dokázal řadit i duplicitní prvky. Pořadí u duplicitních prvků je určeno dle indexu porovnávaných identických prvků a adekvátně inkrementován registr C. Tato modifikace vychází z algoritmu popsaného v [3].

2.1 Algoritmus

1. Všechny registry C nastav na hodnotu 1.
2. Následující činnosti opakuj $2n$ krát, kde $1 \leq k \leq 2n$.
   - Pokud není vstup vyčerpán, vstupní prvek x_i se vloží přes sběrnici do registru X_i a pomocí lineárního spojení do registru Y_1; obsah všech registrů Y se posune doprava.
   - Každý procesor s neprázdnými registry X a Y je porovná, a je-li $X > Y$ inkrementuje svůj registr C. Dále pokud $X = Y$, porovnej index procesoru s indexem prvku v registru Y a pokud i_X[1] $< i_Y$ inkrementuj C.
   - Je-li $k > n$ (po vyčerpání vstupu) procesor P_{k-n} pošle sběrnicí obsah svého registru X procesoru $P_{C_{k-n}}$, který jej uloží do svého registru Z.
3. V následujících n cyklech procesory posouvají obsah svých registrů Z doprava a procesor P_n produkuje seřazenou posloupnost.

[1] Pod indexem čísla v registru X rozumějme rank procesoru.

2.2 Analýza algoritmu

Časová analýza jednotlivých kroků algoritmu: *Krok 1* proběhne v konstatním čase, *Krok 2* trvá $2n$ kroků, *Krok 3* trvá n kroků.

Z těchto poznatků plyne $t(n) = \mathcal{O}(n)$ a k výpočtu potřebujeme n procesorů ($p(n) = n$). Celková cena algoritmu je tedy $c(n) = \mathcal{O}(n^2)$, což značí, že algoritmus enumeration sort není optimální.

3 Implementace

Při inicializaci program vytvoří $n+1$ procesorů, kde procesor P_0 řídí vstup a výstup programu, rozesílá načtená čísla výpočetním procesorům a vypisuje je.

Po načtení čísla x_k velikosti 1 bajt ze souboru `numbers` procesorem P_0 je číslo vypsáno na standardní výstup oddělené mezerou. Dále je číslo sběrnicí posláno procesoru P_{k+1}, uloženo do registru X a je také lineárním spojením posláno procesoru P_1 a uloženo do registru Y. Takto procesor P_0 zpracuje celý soubor.

Procesory $P_1 ... P_{n+1}$ jsou výpočetní procesory. Na sběrnici očekávají jedno číslo x a pokud přijmou číslo y přes lineární spojení, vyjmou ze svého registru Y číslo, odešlou jej svému sousedovi P_{i+1} (vyjma procesoru P_{n+1}, ten číslo y zahodí) a do registru Y zapíšou nové číslo y. Sběrnice i lineární spojení je realizováno pomocí zasílání zpráv.

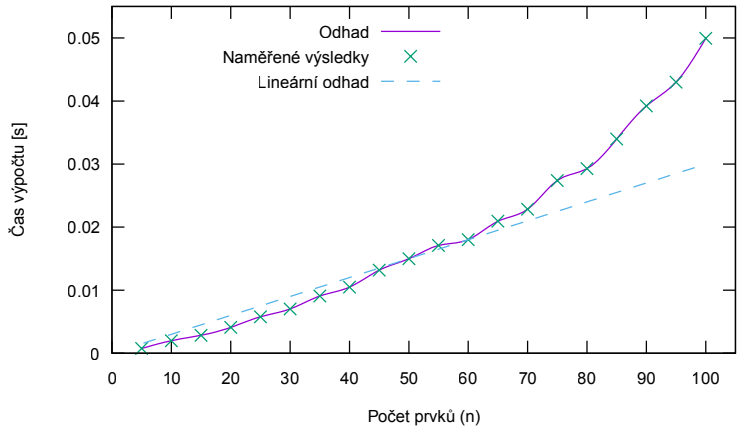
Pokud má procesor neprázdné registry X a Y, poté při každém přijetí nového čísla y provede porovnání registru X a Y a adekvátně inkrementuje registr C. Následně zkontroluje rovnost těchto čísel. Pokud se rovnají, provede kontrolu interně vedeného čítače duplicit čísla x. Pokud je těchto duplicit > 1, porovná index čísla y a rank procesoru a upraví registr C dle algoritmu. Index čísla y je zasílán společně s číslem y.

Po dokončení čtení a rozeslání všech čísel procesorem P_0 a po přenesení všech čísel skrze lineární spojení jsou asynchronně po sběrnici rozeslány čísla x procesoru s rankem odpovídající hodnotě c. Procesory toto číslo uloží do registru Z.

Následně jsou čísla z registru poslána svému sousedovi s nižším rankem. Procesor P_0 je tiskne na výstup, dokud není vytištěno n čísel. Následuje finalizace programu a ukončení.

4 Experimentální ověření časové složitosti

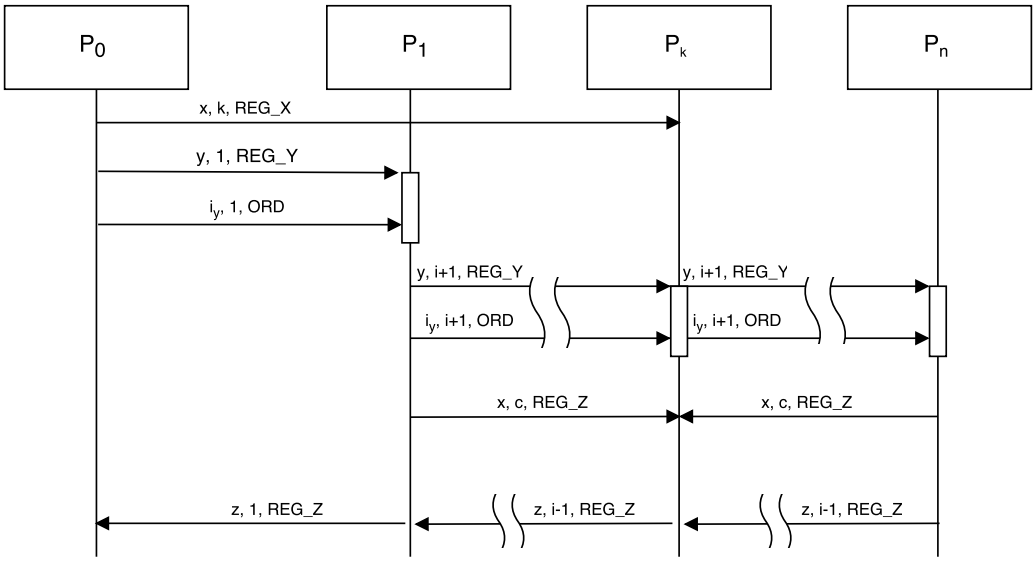
Experimety probíhaly na stroji disponujícím Intel Core i7-2635QM @ 2.00GHz, 8 GB RAM a SSD. Výsledný čas je průměr z 10 měření každého testu. Čas výpočtu byl měřen pomocí Open MPI funkce `Wtime()`. Z naměřených výsledků je patrné, že do 70 procesorů je výpočet lineární. Při větších počtech procesorů se již projevuje režie fyzického procesoru.



Obrázek 2: Experimentálně naměřené výsledky

5 Komunikační protokol

Protokol je znázorněn na obrázku 3. Zasílání zpráv je realizováno výhradně funkcemi `Send`, `Isend`, `Recv` a `Irecv` z knihovny Open MPI. Tag `ORD` je pro zasílání indexu čísla Y, ostatní tagy slouží k zasílání daných hodnot registrům $X/Y/Z$.



Obrázek 3: Sekvenční diagram komunikačního protokolu. Popisky obsahují následující informace: obsah registru, rank, tag.

6 Závěr

Algoritmus se podařilo úspěšně implementovat a experimentálně otestovat. Algoritmus byl upraven tak, aby řadil i duplicitní hodnoty. Z naměřených výsledků lze usoudit, že algoritmus má lineární časovou složitost. Výsledky s více než 70 prvky těmto předpokládům neodpovídají kvůli zvýšené režii při přepínání procesů.

Reference

[1] *Materiály k předmětu paralelní a distribuované algoritmy* [online]. 2017 Dostupné z: <`https://www.fit.vutbr.cz/study/courses/PDA/private/pda.htm`>.

[2] *Open MPI: Open Source High Performance Computing* [online]. 2017 Dostupné z: <`https://www.open-mpi.org`>.

[3] Yasuura, H.; Takagi, N.; Yajima, S. : The Parallel Enumeration Sorting Scheme for VLSI. *IEEE Transactions on Computers*, roč. 31, č. 12, 1982.