Paralelní a distribuované algoritmy
Dokumentácia projektu č. 2

Ján Jusko, Vysoké Učení Technické v Brně 05/04/2017

1 Zadanie

Cieľom tohto projektu je implementovať algoritmus *Enumeration Sort* s lineárnou topológiou za pomoci knihovny *Open MPI*. Program je riadený testovacím skriptom *test.sh* ktorý preloží strojový súbor *es.cpp*, vygeneruje postupnosť náhodných čísel o zadanej dĺžke do súboru *numbers* a následne zavolá program *es*. Pôvodný algoritmus je modifikovaný pridaním jedného riadiaceho procesu takže program pri zoraďovaní *N* prvkov používa *N+1* procesorov.
Program navyše podporuje radenie čísel kde sa môžu vyskytovať duplikáty.

2 Algoritmus

Enumeration Sort je paralelný radiaci algoritmus pracující na lineárnej architektúre, doplnenou o zbernicu, ktorá spája navzájom všetky procesory. Tá je schopná preniesť v každom kroku jednu hodnotu.
Každý procesor P_i má 4 registry:

- **X_i** - prvok vstupnej radenej postupnosti x_i
- **Y_i** - postupne prvky x_1, x_2 ... x_n
- **C_i** - počet prvkov mensích ako x_i
- **Z_i** - zoradený prvok Y_i

2.1 Princíp

1. Všetky registre inicializuj na hodnotu 1.
2. Opakuj $2 * n$ krát, pre $1 \leq k \leq 2 * n$:
   - Pokiaľ nie je vyčerpaný vstup, prvok x_i vlož do registra X_i pomocou zbernice a do registra Y_i pomocou lineárneho posunu doprava cez registre.
   - Každý procesor s neprázdnymi registrami X a Y porovnáva tieto dva hodnoty a v prípade $X > Y$ inkrementuje register svoj C.
   - Po vyčerpaní vstupu, teda $(k > n)$, procesor P_{k-n} pošle po zbernici obsah svojho registru X procesoru P_{Ck-n}, ktorý ho uloží do svojho registra Z.
3. V nasledujúcich n cykloch procesory posúvajú obsah svojich registrov doprava a procesor P_n (posledný) produkuje zoradenú postupnosť.

2.2 Analýza

Celková časová zložitosť algoritmu je daná súčtom časových zložitostí v jednotlivých podkrokoch uvedených vyššie.

- $\Theta(1)$ - inicializácia registrov C
- $\Theta(2 * n)$ - propagácia hodnôt a ich porovnávanie
- $\Theta(n)$ - výpis hodnôt

Časová zloťitosť teda:

$$t(n) = \Theta(1) + \Theta(2 * n) + \Theta(n) = \Theta(n)$$

Cena algoritmu:

$$c(n) = p(n) * t(n) = \Theta(n^2)$$

3 Implementácia

V porovnaní s pôvodným algoritmom bolo nutné vykonať niekoľko modifikácií. Prvou z nich bolo indexovanie procesorov a prvkov radenej postupnosti od čísla 0 vzhľadom na architektúru použitého jazyka *C++*. Ďalšou z nich bola úprava algoritmu k podpore radeniu duplicitných hodnôt. V tejto implementácii pracuje algoritmus interne s hodnotami typu *FLOAT*, pričom pri výskyte duplicitných vstupných hodnôt typu *INT* sa pretypujú a pripočíta sa im veľmi malá hodnota násobená poradím výskytu. Týmto spôsobom nebudú žiadne 2 hodnoty rovnaké. Pri výpise sa použije iba celá časť čísla. Poslednou modofikáciou bolo pridanie jedného pomocného procesoru *n+1* ktorý rieši počiatočný prenos hodnôt, zber a výpis zoradenej postupnosti.

Pomocou knižnice *OpenMPI* program simuluje lineárnu topológiu so spoločnou zbernicou. Lineárne prepojenie je implementované pomocou blokujúcich funkcií *MPI_Send* a *MPI_Recv* ktoré umožnujú odosielať a prijímať hodnoty medzi procesormi. Príjem sa nezaobíde bez udaného príznaku ktorý určuje, ktorá hodnota sa prijíma. V programe boli takto definované flagy pre každý register (napríklad pre register *X* = *REG_X_FLAG*).

Komunikácia medzi procesormi je znázornená na obrázku 1.

4 Výsledok

Implementovaný program pracuje správne podľa zadania aj s duplicitnými hodnotami. Pre meranie výpočetného času behu programu nad rôznym počtom vstupných hodnôt som použil bash funkciu *time*. Priemerné hodnoty sú uvedené v tabuľke 1.

Počet prvkov	Priemerný systémový čas [s]
5	0.116
10	0.192
20	0.392
30	0.460
50	0.956

Table 1: Tabuľka priemerných časových hodnôt

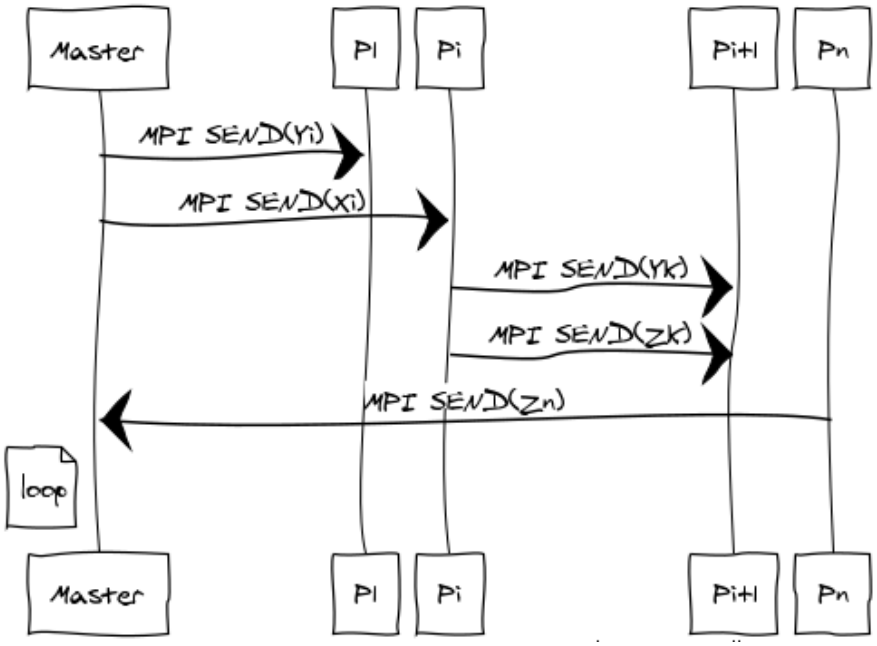

Figure 1: Sekvenčný diagram komunikácie

5 Záver

Z tabulky je očividné, že zo vzrastajúcim počtom radených hodnôt rastie aj čas a to lineárne. To odpovedá teoretickej časovej zložitosti algoritmu. Ostáva len konštatovať, že pre štúdijné účely bol algoritmus úspešne implementovaný.