Paralelní a distribuované algoritmy (2020/2021)
Pipeline Merge Sort

Ondrej Šajdík (xsajdi01)

9. apríla 2021

1 Úvod

Cieľom projektu bolo implementovať Pipeline merge sort na lineárnom poli o n procesoroch, za použitia jazyka *C/C++* a pomocou knižnice *Open MPI*.

2 Rozbor a analýza algoritmu

Pipeline merge sort je paralelný radiaci algoritmus. Procesory sú zapojené lineárne za sebou. Spojené sú s predchodcom a následníkom pomocou dvoch liniek. Na linky posiela postupnosti hodnôt. Posledný procesor má len jednu výstupnú linku, do ktorej posiela už zoradenú postupnosť a prvý procesor má len jednu vstupnú linku so vstupnou postupnosťou. Pre vstup o veľkosti N je potreba $\log_2(\mathrm{N})+1 = \mathrm{k}$ procesorov. Procesory pracujú následovne:

- **Prvý procesor** (P_0) číta vstup a načítané hodnoty posiela ďalšiemu procesoru cez linku. Linky po každej hodnote strieda.
- **Stredné procesory** (P_1 až $\mathrm{P}_{\mathrm{k-1}}$) spájajú postupnosti o veľkosti $2_{\mathrm{i-1}}$ do jednej. Procesor na začiatku čaká, dokiaľ nie je načítaná prvá postupnosť a prvý prvok druhej postupnosti. Následovne porovnáva prvé prvky postupností a odosiela menší ďalšiemu procesoru. Po odoslaní všetkých prvkov jednej postupnosti prestane porovnávať a pošle zvyšné prvky druhej postupnosti. Zmení linku, na ktorú posiela výstup a začne spájať ďalšie dve postupnosti.
- **Posledný procesor** (P_k) pracuje rovnako ako stredné procesory, ale má len jednu linku na ktorú posiela už zoradenú postupnosť.

2.1 Analýza zložitosti

- **Časová zložitosť $\mathbf{t(n) = O(n)}$**. Procesor P_i začína po načítaní $2^{\mathrm{i-1}}+1$ hodnôt. Tzn. že čaká než ich procesor $\mathrm{P}_{\mathrm{i-1}}$ toľko spracuje. Trvá to $2^{\mathrm{i-1}}+1$ cyklov. Z toho plynie, že procesor P_i začína v cykle: $2^i + i$. Dĺžka radenia

jedného procesoru po začatí je n cyklov, pretože v každom cykle spracuje práve jednu hodnotu. Algoritmus je ukončený, po spracovaní posledného prvku posledným procesorom. Takže algoritmus skončí po 2^k+k+n = $2^{\log_2 n}$ + $\log_2$n + n . Z čoho plynie časová zložitosť O(n)

- **Priestorová zložitosť p(n) = O(log(n))**. Je daná počtom potrebných procesorov, tzn. log(n)+1
- **Celková cena c(n) = O(n log n)**. algoritmu je vypočítaná vynásobením časovej zložitosti počtom procesorov, teda: c(n) = t(n)*p(n)=O(n)*($\log_2$n+1) = O(n log n). Algoritmus je optimálny, pretože cena sa rovná cene optimálneho sekvenčného algoritmu.

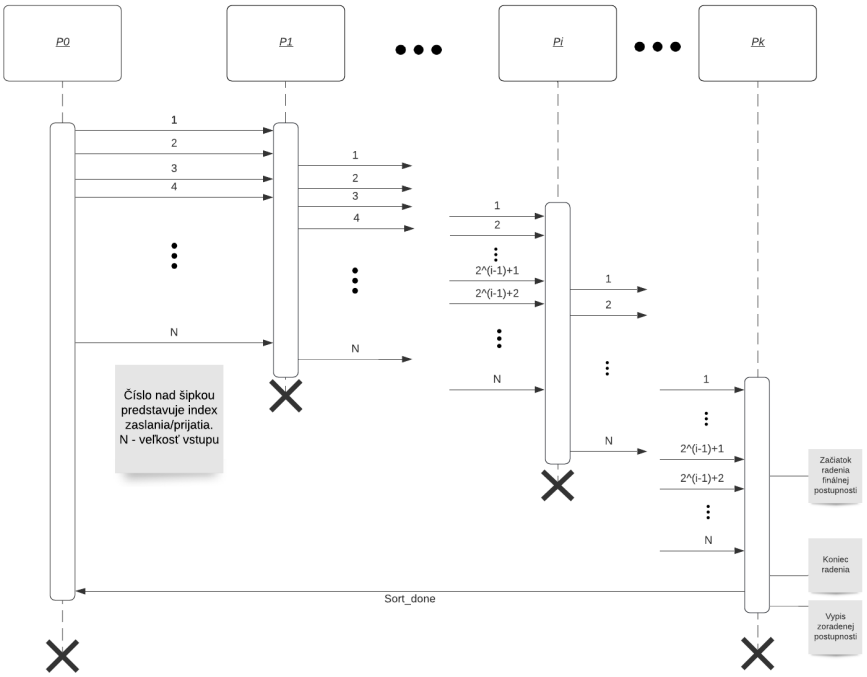
3 Implementácia

Na rozdiel od popisu algoritmu, v implementácii nie sú rozdeľované hodnoty do liniek odosielateľom, ale prijímateľom. Odosielateľ pri odosielaní nerozlišuje do ktorej linky prvok patrí, ale prijímateľ si ich rozdeľuje sám. Ukladá ich do dvoch štruktúr, kde každá predstavuje jednu linku. Je použitá štruktúra *std::queue* , pretože sa chová rovnako ako linky popísané v algoritme. Inak je implementácia algoritmu rozdelená do 3 častí:

1. **Prvý procesor** vo for cykle načíta vstup a posiela ho pomocou funkcie *MPI_Send()* po jednom ďalšiemu procesoru.
2. **Stredný procesor** vykonáva svoju činnosť v dvoch častiach. V prvej iba načítava vstup pomocou *MPI_Recv()* a ukladá si ho. Keď je načítaných dostatok hodnôt prechádza do druhej časti. Tu pracuje N cyklov, kde v každom cykle odošle jednu hodnotu ďalšiemu procesu. Zároveň v každom cykle načítava ďalšie hodnoty, dokiaľ nie je načítaných všetkých N hodnôt.
3. **Posledný procesor** pracuje takmer rovnako ako stredný procesor, ale namiesto posielania hodnôt ďalšiemu procesoru ich vypisuje na štandardný výstup.

4 Komunikačný protokol

Procesy si posielajú hodnoty cez správy. Pri odosielaní sa nerieši do ktorej linky hodnota patrí, ale rozdelenie do liniek je vykonané príjemcom.

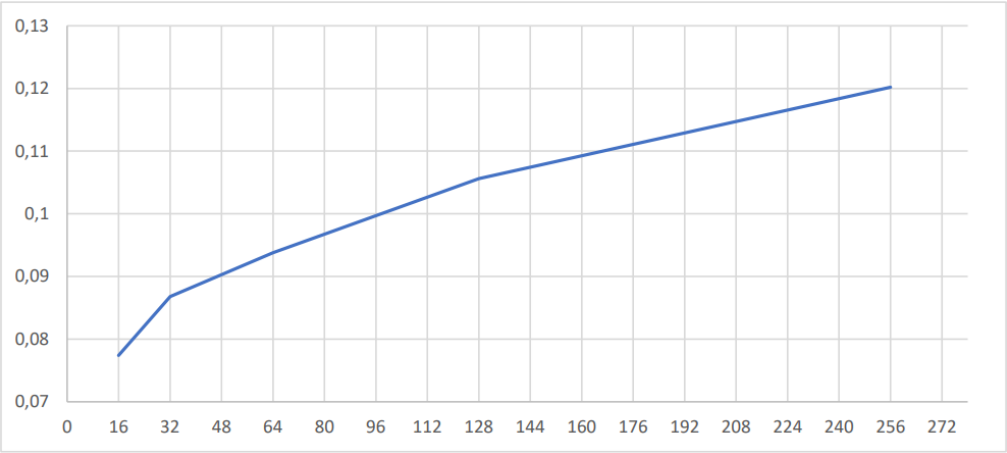


Obr. 1: Sekvenčný diagram

5 Experimenty

Implementácia algoritmu bola testovaná pre rôzne veľké vstupy pre overenie časovej zložitosti. Pre meranie bol použitý nástroj *time* a bolo vykonané na servere *merlin*. Sledovaný bol čas *sys*. Meralo sa veľkosti vstupov: 16,32,64,128,256. Každý počet vstupov bol meraný 10-krát a výsledok spriemerovaný.Pre každé meranie bol vygenerovaný nový vstupný súbor. Namerané hodnoty môžme vidieť v tabuľke a v grafe.

Veľkosť vstupu	**Nameraný čas (priemer)**
16	0,0774
32	0,0868
64	0,0938
128	0,1056
256	0,1202



Obr. 2: Graf s priemernými výsledkami

6 Záver

Z nameraných dát a z grafu na obrázku 2 môžme vidieť, že s rastom veľkosti vstupu rastie čas potrebný pre vykonanie implementovaného algoritmu viac menej lineárne. Prípadné výkyvy môžu byť spôsobené malým množstvom meraní alebo nestálosťou prostredia.