Implementácia algoritmu "Bucket sort"

Martin Riša

22. března 2019

1 Rozbor a analýza algoritmu

Zoraďovanie vyváženým stromom procesorov s listovými procesormi $n = 2^m$. Strom obsahuje $2m - 1$ procesorov, takže $p(n) = (2.log(n)) - 1$. Každý listový procesor obsahuje n/m zoraďovaých prvkov a zoraďuje ju optimálním sekvenčným algoritmom $\mathcal{O}(n.log(n))$, kde n je veľkosť bucketu listového uzlu. Každý nelistový procesor vie spojiť dve zoradené postupnosti optimálním sekvenčním algoritmem $\mathcal{O}(n)$, kde n je veľkosť bucketu daného uzlu.

Predošlé je pravda, keď veľkosť vstupu je rovná mocnine dvojky, v opačnom prípade je vstup doplnený na najbližšiu mocninu dvojky vzhľadom na požadovanú veľkosť stromu, tak aby bol strom vyvážený.

- Koreňový uzol náčítava celý vstup teda $\mathcal{O}(n)$.
- Každý listový procesor číta $n/log(n)$ vstupných hodnôt teda $\mathcal{O}(n/log(n))$.
- Každý listový procesor zoraďuje bucket o veľkosti $log(n/log(n))$ teda $\mathcal{O}(n/log(n).log(n/(log(n)))) = \mathcal{O}n$.
- Pri j-té iterácie každý procesor na úrovni $i = log(m) - j$ spojí dve postupnosti o dĺžke $n/2^i$. Je použitý straight merge, ktorému každá iterácia trvá $k.n/2^i$, teda $\sum_{i=1}^{log(m)-1}(k.n)/2^i = \mathcal{O}(n)$.
- Ukladanie výsledku zabere $\mathcal{O}(n)$.

Výsledne, je algoritmus optimálny:

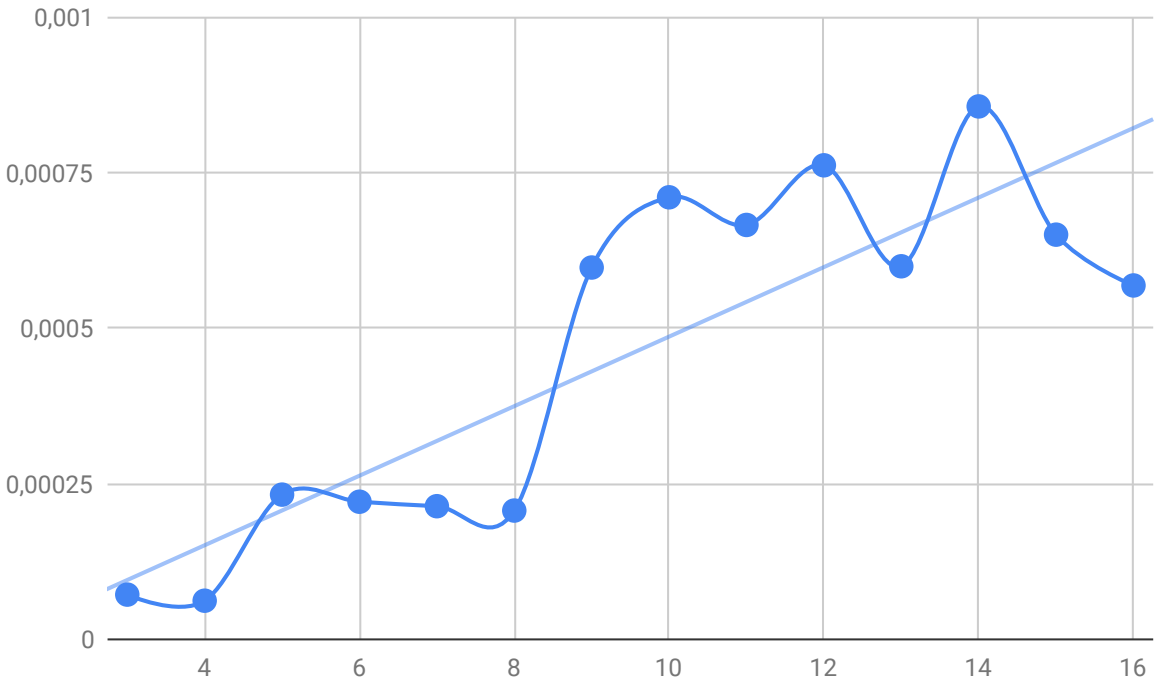
- $t(n) = \mathcal{O}(n)$
- $p(n) = \mathcal{O}(log(n))$
- $c(n) = \mathcal{O}(n.log(n))$

2 Implementácia

Implementovaný je algoritmus **Bucket sort** vo variante kde koreňový uzol načítava vstupné dáta, ktoré sú rozdelené do **bucket-ov** a rozposlané jednotlivým listovým uzlom. Algoritmus na začiatku načíta vstup a ak počet vstupov nie je najbližšou vyššou mocninou dvojky vzhľadom na požadovanú výšku stromu, tak sú vstupné dáta doplnené hodnotami **255**. Zvyšok algoritmu je totožný s tým uvedeným na prednáške.

3 Experimenty

... sa vykonávali na stroji **merlin.fit.vutbr.cz**. Nakoľko má každý študent obmedzenie počtu bežiacich procesov na 60, nepodarilo sa mi experimentami overiť časovú zložitosť. Na grafe sú znázornené experimenty pre 3 až 16 vstupných hodnôt. Na grafe je vidieť, že najefektívnejším je zoraďovanie vstupných dát keď je ich počet rovný mocnín dvojky, čo sa dalo očakávať.



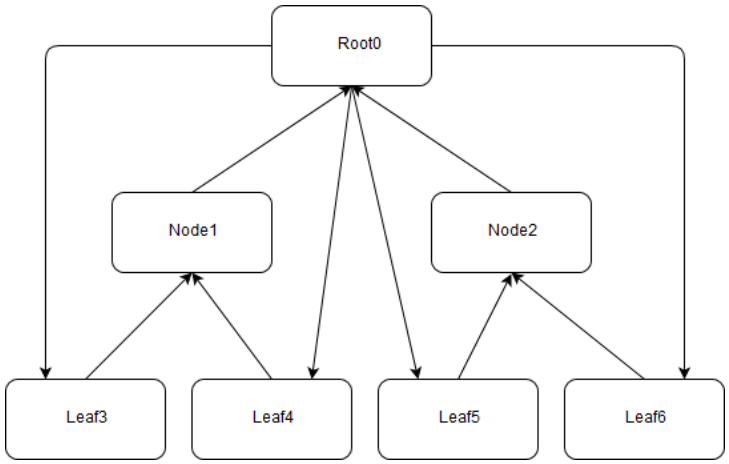
Obrázek 1: Graf experimentov pre rôzne veľké vstupy

4 Komunikačný protokol

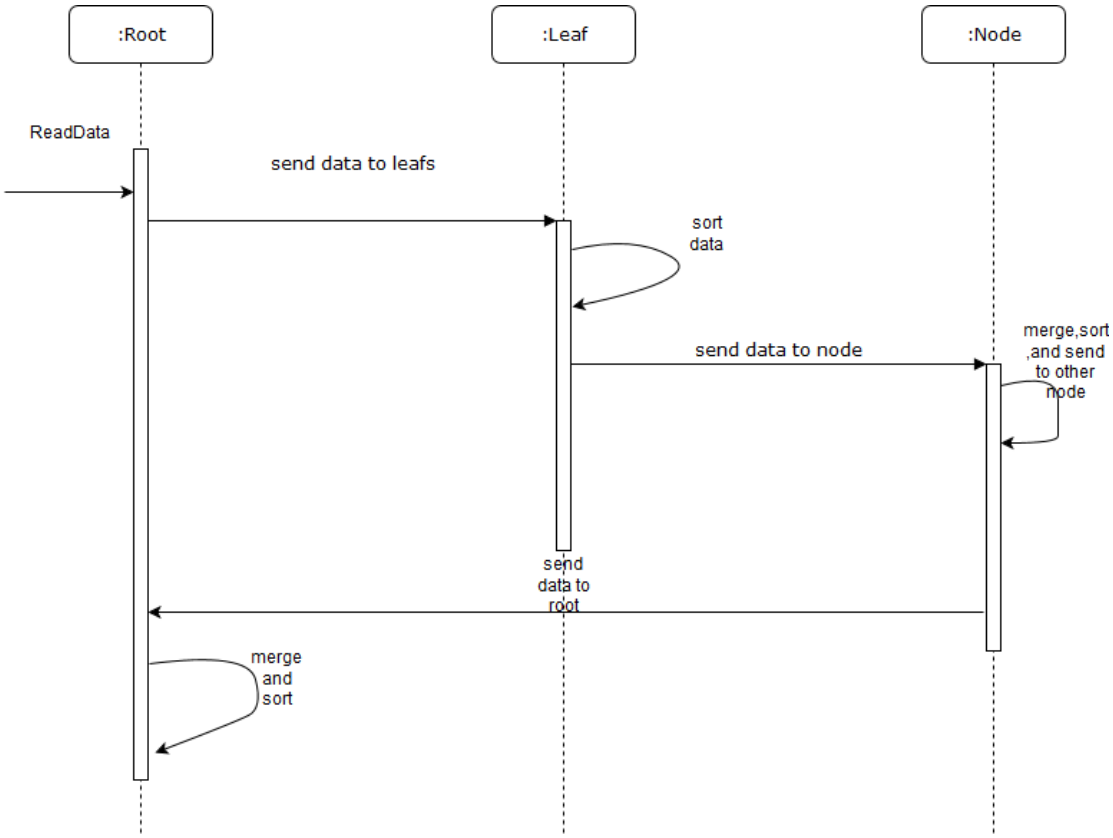
Koreňový uzol načíta data, ktoré rozdelí do `bucket-ov` a rozpošle listovým uzlom. Tie vstup zoradia konvenčným sekvenčným algoritmom na zoraďovanie a odošlú nadradenému uzlu informáciu o veľkosti odosielaných dát nasledované samotnými dátami.

5 Záver

Časovú zložitosť sa mi overiť nepodarilo, nakoľko som nemal k dispozícií stroj, na ktorom by som mohol prekročiť hranicu podprocesov nad 60. Z nameraných hodnôt vyplýva, že implementvaný algoritmus najrýchlejšie zoraďuje vstup keď jeho veľkosť je mocninou dvojky.



Obrázek 2: Komunikace pre 7 procesov spolu s číslovaním



Obrázek 3: Sekvenčný diagram komunikácie