Distribuované a paralelní algoritmy - algoritmy řazení, select

Z FITwiki

Vše vypracováno zde: https://fituska.eu/viewtopic.php?p=306919#p306919

Distribuované a paralelní algoritmy a jejich složitost, algoritmy řazení, select - přednáška (https://www.fit.vutbr.cz/study/courses/PDA/private/www/h003.pdf)

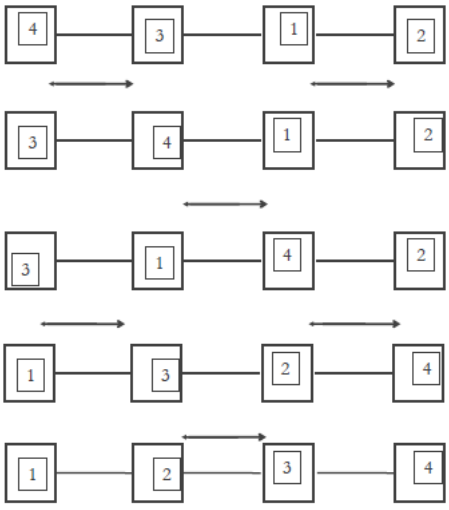
Obsah

Enumeration sort

- Princip: výsledná pozice prvku je dána počtem prvků, které jsou menší
  - Ideální algoritmus pro paralelní zpracování (ale drahej)
- Topologie: mřížka n × n
- Časová složitost: $t(n) = O(log(n))$
- Počet procesorů: $p(n) = O(n^2)$
- Cena: $c(n) = O(n^2.log(n))$; není optimální
- Algoritmus (zjednodušeně):
  - 1. krok - distribuce hodnot na procesory (v řádcích i sloupcích). Složitost kroku O(log n)
  - 2. krok - porovnání hodnot a sčítání ve stromové struktuře. Složitost O(log n)
  - 3. krok - předání hodnot na správné pozice (procesory)

Odd-even transposition sort

- Princip: paralelní bubble-sort, porovnávají se jen sousedé a mohou se přehodit
- Topologie: lineární pole n procesorů
- Časová složitost: $t(n) = O(n)$
- Počet procesorů: $p(n) = O(n)$
- Cena: $c(n) = O(n^2)$; není optimální
- Algoritmus:
  - V prvním kroku se každý lichý procesor p i spojí se svým sousedem p i+1 a porovnají své hodnoty je-li y i >y i+1, pak vymění své hodnoty
  - V druhém kroku se každý sudý procesor ...totéž...
  - Po n krocích (maximálně) jsou hodnoty seřazeny



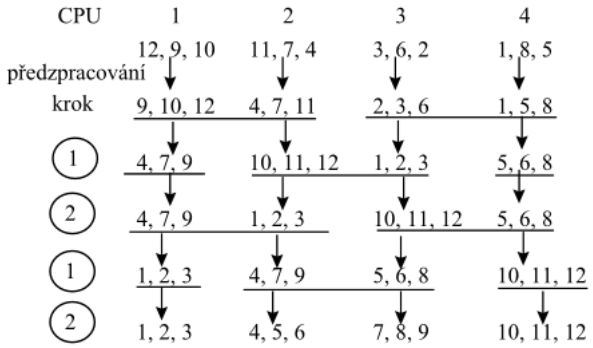
Odd-even merge sort

- Princip: Řadí se speciální sítí procesorů (Každý procesor má dva vstupní a dva výstupní kanály)
  - Každý procesor umí porovnat hodnoty na svých vstupech, menší dá na výstup L(low), a větší dá výstup H (high)
- Topologie: sítě procesorů 1×1, 2×2, 4×4, 8×8, ... (procesory propojeny tak, aby složením jednotlivých porovnání byla seřazená posloupnost)

- Časová složitost: $t(n) = O(log^2(n))$
- Počet procesorů: $p(n) = O(n * log^2(n))$
- Cena: $c(n) = O(n.log^4(n))$; není optimální

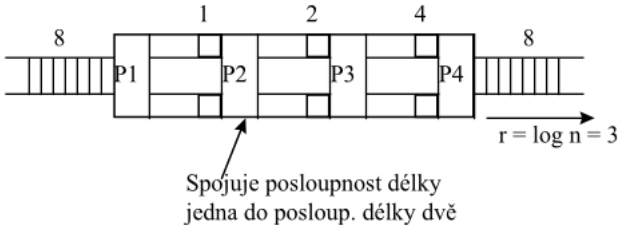
Merge-splitting sort

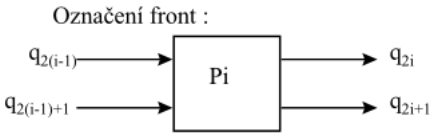
- Princip: varianta odd-even sortu, každý procesor řadí krátkou posloupnost
  - Porovnání a výměna je nahrazena operacemi merge-split
- Topologie: lineární pole procesorů
- Časová složitost: $t(n) = O(n.log(n) / p) + O(n)$
- Počet procesorů: $p(n) < n$
- Cena: $c(n) = O(n.log(n)) + O(n.p)$; optimální pro $p \leq log(n)$
- Algoritmus (zjednodušeně):
  - Místo porovnání sousedů se provede spojení posloupností (O(n)) a pak rozdělení na půl



Pipeline merge sort

- Princip: rozděleno na několik kroků, první spojuje posloupnosti délky 1, pak 2, atd.
  - Data nejsou uložena v procesorech, ale postupně do nich vstupují
  - Každý procesor spojuje dvě seřazené posloupnosti délky 2^{i-2}
- Topologie: lineární pole procesorů
- Časová složitost: $t(n) = O(n)$
- Počet procesorů: $p(n) = log(n) + 1$
- Cena: $c(n) = O(n).O(log(n) + 1) = O(n.log(n))$; optimální

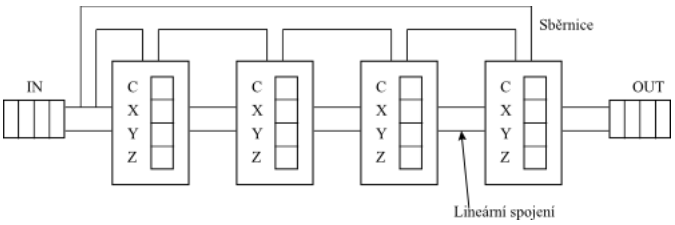




Enumeration sort

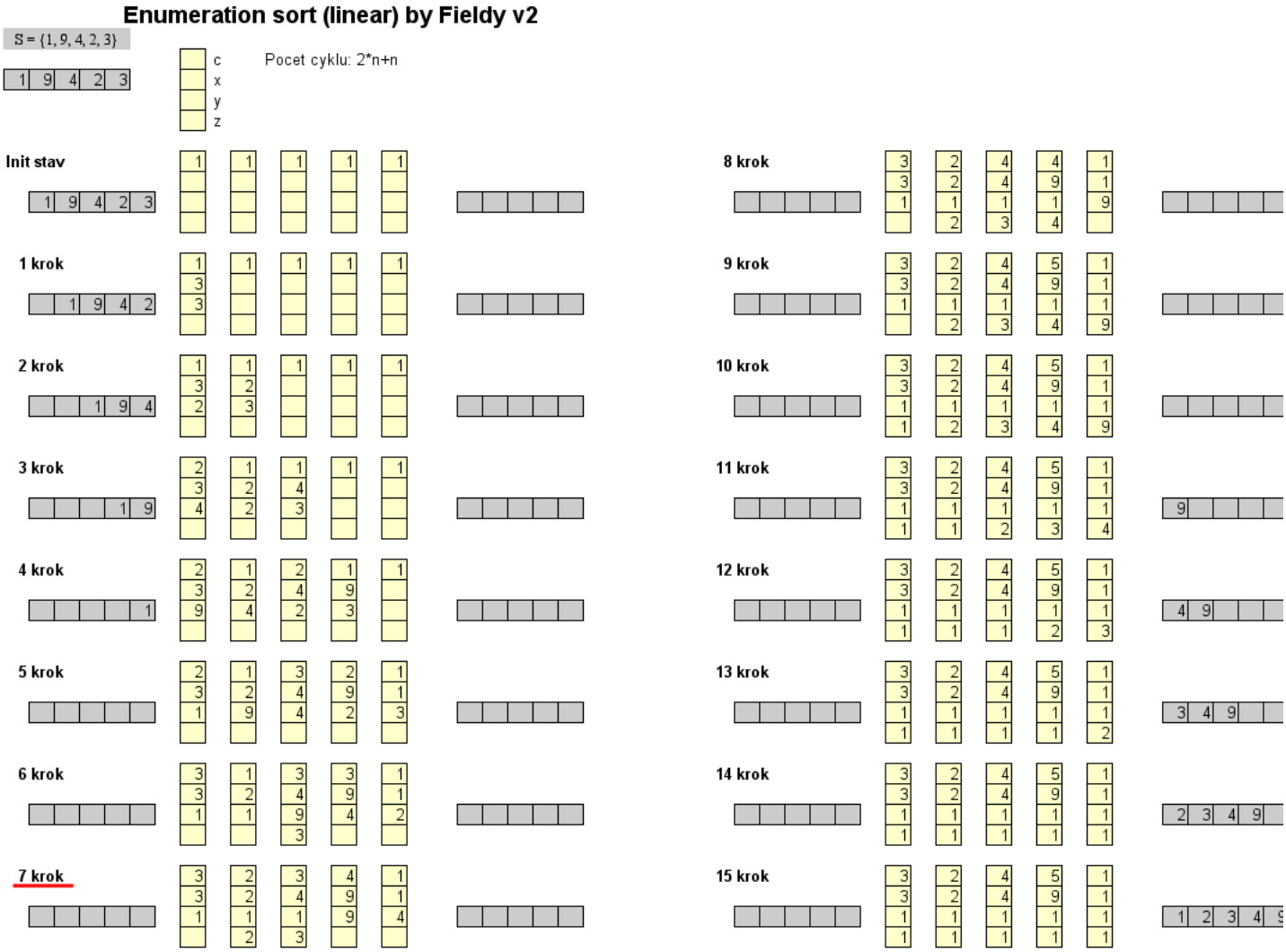
- Princip: procesorem každého prvku proteče celá posloupnost, tím zjistí pořadí svého prvku a po sdílené sběrnici jej pošle na odpovídající výstup.
- Topologie: lineární pole procesorů a sdílená sběrnice, která muže přenést jednu hodnotu v každém kroku
- Časová složitost: $t(n) = O(n)$
- Počet procesorů: $p(n) = n$
- Cena: $c(n) = O(n^2)$; není optimální

Všechny prvky se nechají protéct všemi procesory. Procesory přitom počítají počet prvků menších než jejich prvek. Po protečení všech prvků tak procesor zná pořadí svého prvku - odešle prvek boční sdílenou sběrnicí odpovídajícímu výstupnímu procesoru. (Na sdílené sběrnici vysílá vždy jen procesor kterému právě odtekl konec posloupnosti - nekolidují.)



Legenda:

- C_i počet prvků menších než x_i (t.j. kolikrát byl $Y_i \leq X_i$)
- X_i prvek x_i
- Y_i postupně prvky $x_1 ... x_n$
- Z_i seřazený prvek Y_i



Minimum extraction sort

- Princip: stromem odebírá vždy nejmenší prvek
  - Každý listový procesor obsahuje jeden řazený prvek
  - Každý nelistový procesor umí porovnat dva prvky
- Topologie: strom
- Časová složitost: $t(n) = O(n)$
- Počet procesorů: $p(n) = 2n - 1$
- Cena: $c(n) = O(n^2)$; není optimální

Bucket sort

- Princip: stromem spojené procesory, které řadí menší posloupnosti a pak spojení
  - Každý listový procesor obsahuje n/m řazených prvků a umí je seřadit optimálním sekvenčním algoritmem (např. heapsort)
  - Každý nelistový procesor umí spojit dvě seřazené posloupnosti optimálním sekvenčním algoritmem
- Topologie: strom
- Časová složitost: $t(n) = O(n)$
- Počet procesorů: $p(n) = 2.log(n) - 1$
- Cena: $c(n) = O(n.log(n))$; optimální

Median finding and splitting

- Princip: dělí neseřazenou posloupnost podle mediánu mezi potomky, jednotlivé listy své posloupnosti nakonec seřadí sekvenčním sortem
  - Každý list umí optimální sekvenční sort
  - Každý nelistový procesor umí nalézt medián v optimálním čase (např. algoritmus Select s O(n) složitostí)
  - Ekvivalent Quick Sortu - liší se tím, že se jednotlivé stupně provádí paralelně.
- Topologie: strom
- Časová složitost: $t(n) = O(n)$
- Počet procesorů: $p(n) = 2.log(n) - 1$
- Cena: $c(n) = O(n.log(n))$; optimální
- Algoritmus (zjednodušene):
  - Nelistový procesor najde medián, rozdělí posloupnost a předá ji níže
  - Až dojde posloupnost do listového procesoru, dojde k jejímu seřazení
  - V listech pak máme postupně celou seřazenou posloupnost

Select

Sequential select

- Princip: hledá k-tý nejmenší prvek v posloupnosti S
  - je-li k=|S|/2, jde o medián
- Algoritmus (zjednodušeně):
  - Rozdělíme vstupní posloupnost na několik pod posloupností, ty seřadíme, najdeme v každé medián a ten vrátíme
  - Dostaneme posloupnost mediánu, tu seřadíme a najdeme v ní medián
  - Původní vstupní posloupnost rozdělíme na tři části podle nalezeného "mediánu mediánů" (L - mensi, E - rovno, G - vetsi)
    - (k je délka vstupní posloupnosti dělena 2)
    - Pokud |L| > k - medián musí být v L - Zavolej algoritmus znovu tentokrát pro posloupnost L
    - Pokud |L| + |E| > k - medián musí být v E - Vrať nalezený medián
    - Jinak - medián musí být v G - Zavolej algoritmus znovu tentokrát pro posloupnost G

Parallel select

- Princip: k-tý nejmenší prvek v posloupnosti S; EREW PRAM s N procesory P1..Pn; používá sdílené pole M o N prvcích
- Složitost: t(n) = O(n/N) pro n > 4, N < n/log n; p(n) = N; c(n) = t(n).p(n) = O(n) - optimální

- *Problém, který degraduje složitost algoritmus je, že při zpracování posloupností L, E, G nevíme, kam uložit číslo do paměti, dokud tam neuložíme všechny předchozí. Toto řeší následující algoritmus Parallel splitting*

Parallel splitting

- Využívá se jako součást Parallel select (pro složení Li jednotlivých posloupností do jediného L - Krok 4 algoritmu)
- Úkol: Je dána posloupnost S a číslo m; Mají se vytvořit tři posloupnosti:
  - L = {si ε S: si < m}
  - E = {si ε S: si = m}
  - G = {si ε S: si > m}
  - *Po tom co se vytvoří se vypočítá, kam mají procesory ukládat své výsledné posloupnosti (suma prefixů), aby mohly ukládat současně a nemuseli čekat, než budou uloženy všechny předchozí hodnoty.*
- Složitost sekvenčního algoritmu je O(n)
- Paralelní řešení - máme N procesorů, které si sekvenci S rozdělí na podposloupnosti o délce n/N:
  - Každý procesor sestrojí L_i (resp. E_i, G_i) své podposloupnosti
  - Pomocí sumy prefixů jsou vypočítány z_i - pozice kde mají jednotlivé L_i začínat ve výsledném L:

$$z_i = \sum_{j=1}^{i} |L_j|$$

- Složitost: t(n) = O(log N + n/N) = O(n/N) pro dostatečně malé N; Cena c(n) = O(n) - optimální

Kategorie: Paralelní a distribuované algoritmy | Státnice 2013 | Státnice MMI

- Stránka byla naposledy editována 1. 6. 2016 v 22:10.