PARALELNÍ TŘÍDĚNÍ

umožňuje rychlejší řešení

zpracování dat větší velikosti

Paralelní modely výpočtu :

Paralelní počítač - soubor procesorů téhož typu propojených tak, aby byla umožněna koordinace jejich činnosti a výměna dat

Distribuovaný systém - množina procesorů (počítačů) různých typů distribuovaných na velkém prostoru, spojených komunikační sítí

účelem je najít zdroj informací a informace přenášet

nebudeme se jím zabývat

Modely paralelních počítačů

1) DAG - orientovaný acyklický graf

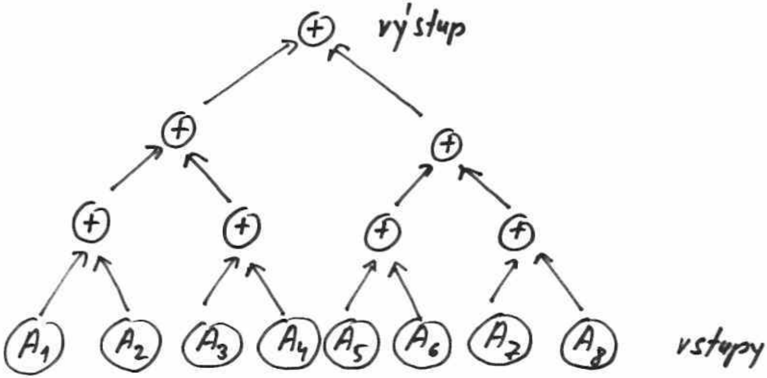
vstupy - uzly, do kterých nevcházejí žádné hrany

vnitřní uzly - operace

výsledek - uzel, z něhož nevede žádná hrana

operace probíhají v jednotkovém čase

Příklad : sčítání 8 čísel



hodí se pro numerické výpočty

neumožňuje větvení a cykly

algoritmus každému uzlu přiřadí procesor -

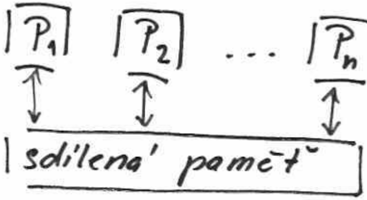
stačí $\frac{n}{2}$ procesorů a $\log n$ času

je nezávislý na použité architektuře

2) Paralelní RAM – PRAM

soubor procesorů typu RAM

komunikují přes sdílenou paměť



činnost je synchronní

každý procesor vykonává tentýž program,
tutéž instrukci v téže časové jednotce

je to model typu SIMD (single instruction
multiple data)

podle přístupu ke sdílené paměti:

EREW, CREW, CRCW

pravidla pro současný zápis: COMMON
priority

3) model se sdílenou pamětí MIMD
(multiple instruction multiple data)
každý procesor má svůj program
komunikace je asynchronní

4) Sítě
graf, kde uzly jsou procesory
hrany jsou spojení mezi nimi
mohou být synchronní i asynchronní
SIMD i MIMD
mají různou topologii: lineární seznam
kruh
síť (mříž)
krychle, hyperkrychle
strom
⋮

Další dělení:
speciální – pro řešení jednoho problému
(např. třídicí sítě)
víceúčelové

Míry složitosti paralelních algoritmů

n … rozsah vstupních dat

$T(n)$ … doba výpočtu (paralelního)

$P(n)$ … počet použitých procesorů

efektivní algoritmy – $P(n) = O(n^k)$

$$T(n) = O(\log^k n)$$

cena paralelního algoritmu : $C(n) = T(n)P(n)$

(paralelní algoritmus se dá převést na sekvenční s časem $O(C(n))$)

další míry :

zrychlení = nejhorší čas nejrychlejšího sekvenčního algoritmu : nejhorší čas paralelního algoritmu

(ideální je podíl N při použití N procesorů)

eficience = nejhorší čas nejrychlejšího sekvenčního algoritmu : cena paralelního algoritmu

(většinou je ≤ 1)

Dolní odhad složitosti paralelního třídění

dolní odhad pro sekvenční algoritmy: $\Omega(n \log n)$

z toho plyne:

$\Omega\left(\frac{n \log n}{N}\right)$ pro paralelní při N procesorech

speciálně:

při n procesorech čas $\Omega(\log n)$

při $\log n$ procesorech čas $\Omega(n)$

čas se měří počtem porovnání

Při sekvenčním vstupu (výstupu) dat nemůže být čas lepší než lineární (při libovolném počtu procesorů).

TŘÍDICÍ SÍTĚ

Batcher 1968

Odd-even Mergesort

Bitonické mergování

procesory = komparátory pro 2 prvky

složitost :

$O(\log^2 n)$ hladin

$O(n \log^2 n)$ komparátorů

nejsou optimální

Ajtai, Kolmós, Szemerédi (1983)

optimální třídicí síť

$O(\log n)$ hladin

$O(n \log n)$ komparátorů

algoritmus je komplikovaný

nepraktický - multiplikativní konstanta je hodně vysoká

SÍTĚ SIMD

1) Lineárně propojené procesory

$\boxed{P_1} - \boxed{P_2} - \boxed{P_3} - \cdots - \boxed{P_r}$

Odd-even transposition sort

Knuth (1973) s odkazem na Demuth (1956)

třídíme posloupnost $x_1, \dots, x_n$

počet procesorů $P(n) = n$

y_i … prvek posloupnosti uchovaný v procesoru P_i

na začátku $y_i = x_i$ pro všechna i

Algoritmus:

for $k := 1$ **to** $\lceil \frac{n}{2} \rceil$ **do**

  for $i := 1, 3, \dots, 2\lceil \frac{n}{2} \rceil - 1$ **do in parallel**

    if $y_i > y_{i+1}$ **then** $y_i \leftrightarrow y_{i+1}$ **endif**

  enddo

  for $i = 2, 4, \dots, 2\lfloor \frac{n-1}{2} \rfloor$ **do in parallel**

    if $y_i > y_{i+1}$ **then** $y_i \leftrightarrow y_{i+1}$ **endif**

  enddo

enddo

<u>Příklad:</u> 7 6 5 4 3 2 1

$k=1$: 6 7 4 5 2 3 1

6 4 7 2 5 1 3

$k=2$: 4 6 2 7 1 5 3

4 2 6 1 7 3 5

$k=3$: 2 4 1 6 3 7 5

2 1 4 3 6 5 7

$k=4$: 1 2 3 4 5 6 7

Důkaz korektnosti – indukcí

<u>Složitost:</u>

čas: $T(n) = \lceil \frac{n}{2} \rceil \cdot 2 \cdot (1 \text{ porovnání} + 1 \text{ výměna}) = O(n)$

počet procesorů: $P(n) = n$

cena: $C(n) = T(n)P(n) = O(n^2)$

není optimální

Merge-splitting sort

Baudet, Stevenson (1978)

$P(n) = p < n$

každý procesor obsahuje $\frac{n}{p}$ prvků posloupnosti

S_i ... prvky uchované v procesoru P_i

Algoritmus:

for $i := 1$ **to** p **do in parallel** sort S_i **enddo**

for $k := 1$ **to** $\lceil \frac{p}{2} \rceil$ **do**

  for $i = 1, 3, \dots, 2\lceil \frac{p}{2} \rceil - 1$ **do in parallel**

    merge S_i a S_{i+1} do uspořádané posloupnosti S_i'

    $S_i \leftarrow$ prvních $\frac{n}{p}$ prvků S_i'

    $S_{i+1} \leftarrow$ druhých $\frac{n}{p}$ prvků S_i'

  enddo

  for $i = 2, 4, \dots, 2\lfloor \frac{p-1}{2} \rfloor$ **do in parallel**

    totéž

  enddo

enddo

Příklad: pro 4 procesory

	12, 9, 10	11, 7, 4	3, 6, 2	1, 8, 5
	9, 10, 12	4, 7, 11	2, 3, 6	1, 5, 8
$k=1$:	4, 7, 9	10, 11, 12	1, 2, 3	5, 6, 8
	4, 7, 9	1, 2, 3	10, 11, 12	5, 6, 8
$k=2$:	1, 2, 3	4, 7, 9	5, 6, 8	10, 11, 12
	1, 2, 3	4, 5, 6	7, 8, 9	10, 11, 12

Složitost:

čas: sekvenční třídění $O(\frac{n}{p} \log \frac{n}{p})$

$$\left.\begin{array}{l} \text{transfer } S_{i+1} \text{ do } P_i : O(\frac{n}{p}) \\ \text{merge } S_i \text{ a } S_{i+1} : O(2 \cdot \frac{n}{p}) \\ \text{transfer do } P_{i+1} : O(\frac{n}{p}) \end{array}\right\} O(\frac{n}{p})$$

$T(n) = O(\frac{n}{p} \log \frac{n}{p}) + O(\frac{p}{2} \cdot \frac{n}{p} \cdot 2) = O(\frac{n}{p} \log \frac{n}{p}) + O(n)$

$P(n) = p$

$c(n) = pT(n) = O(n \log n) + O(np)$

je optimální pro $p \leq \log n$

Mergesort na rouře

Todd 1978

schéma pro posloupnost délky $n = 2^r$

data postupují zleva doprava

P_1 : čte data sekvenčně ze vstupu a posílá je střídavě
na horní a dolní výstup

P_i pro $i = 2, \dots, r$:

dostává 2 posloupnosti délky 2^{i-2}
(každou na jedné vstupní lince)

merguje je do posloupnosti délky 2^{i-1}

začíná mergovat, když má 1 vstup plný a
na druhém 1 prvek

výsledky posílá střídavě na horní a dolní výstup

P_{r+1} : merguje 2 posloupnosti do jedné výsledné

Příklad:

1, 5, 3, 2, 8, 7, 4, 6	P_1	—	P_2	—	P_3	—	P_4	—
1, 5, 3, 2, 8, 7, 4	P_1	6	P_2	—	P_3	—	P_4	—
1, 5, 3, 2, 8, 7	P_1	6 4	P_2	—	P_3	—	P_4	—
1, 5, 3, 2, 8	P_1	7 4	P_2	6	P_3	—	P_4	—
1, 5, 3, 2	P_1	7 8	P_2	4, 6	P_3	—	P_4	—
1, 5, 3	P_1	2, 7	P_2	4, 6 8	P_3	—	P_4	—
1, 5	P_1	2 3	P_2	4, 6 7	P_3	8	P_4	—
1	P_1	5, 2	P_2	3, 4, 6	P_3	7, 8	P_4	—
—	P_1	5 1	P_2	2, 3, 4	P_3	6, 7, 8	P_4	—
—	P_1	1	P_2	2, 3 5	P_3	4, 6, 7, 8	P_4	—
—	P_1	—	P_2	2, 3 1	P_3	4, 6, 7, 8 5	P_4	—

— P_1 = P_2 $\genfrac{}{}{0pt}{}{2}{1}$ P_3 $\genfrac{}{}{0pt}{}{4,\,6,\,7}{3,\,5}$ P_4 8

— P_1 = P_2 — 1 P_3 $\genfrac{}{}{0pt}{}{4,\,6}{2,\,3,\,5}$ P_4 7, 8

— P_1 = P_2 = P_3 $\genfrac{}{}{0pt}{}{4}{1,\,2,\,3,\,5}$ P_4 6, 7, 8

— P_1 = P_2 = P_3 $\genfrac{}{}{0pt}{}{4}{1,\,2,\,3}$ P_4 5, 6, 7, 8

— P_1 = P_2 = P_3 — 1, 2, 3 P_4 4, 5, 6, 7, 8

atd.

— P_1 = P_2 = P_3 = P_4 1, 2, 3, 4, 5, 6, 7, 8

Složitost :

$P(n) = r+1 = O(r) = O(\log n)$

čas: začíná činnosti P_1 (cyklus 1)

končí, když ukončí činnost P_{r+1}

P_i začíná činnost, když jeden jeho vstup má 2^{i-2} prvků a druhý 1 prvek

tj. o $2^{i-2}+1$ cyklů později než P_{i-1}

celkem o $\sum_{j=0}^{i-2} (2^j+1) = 2^{i-1} - 1 + i - 1 = 2^{i-1} + i - 2$

cyklů později než P_1

tj. v cyklu $2^{i-1} + i - 1$

P_{r+1} začne v cyklu $2^r + r$

skončí o $n-1$ cyklů později, tj. v čase $n + 2^r + r - 1 =$

$= 2n + \log n - 1 = O(n)$

$T(n) = O(n)$

$c(n) = O(n \log n)$

je optimální

2) Stromová struktura

strom s d hladinami (binární)

2^{d-1} vrcholy

každý vrchol je procesor

kořen a listy obstarávají vstup a výstup

Selection sort

procesor obsahuje 1 prvek

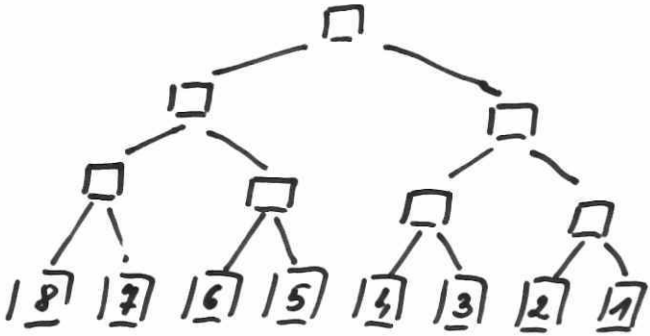
vstupy jsou v listech

vnitřní procesor, který je prázdný, si vezme menší z čísel, která jsou v jeho synech

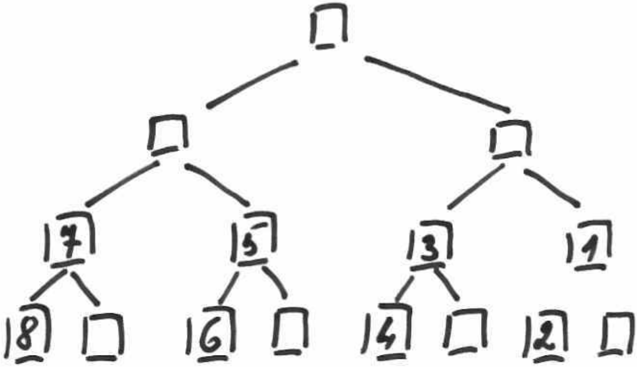
kořen posílá svůj obsah na výstup

Příklad: třídím 8, 7, 6, 5, 4, 3, 2, 1

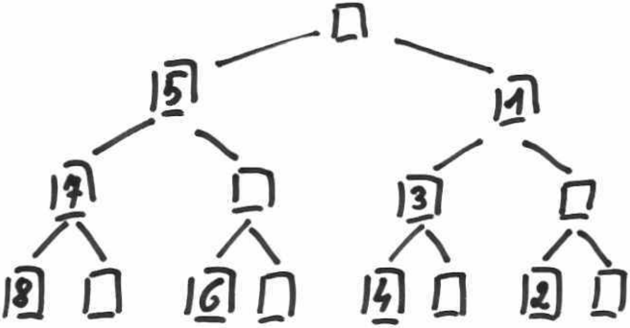
0)



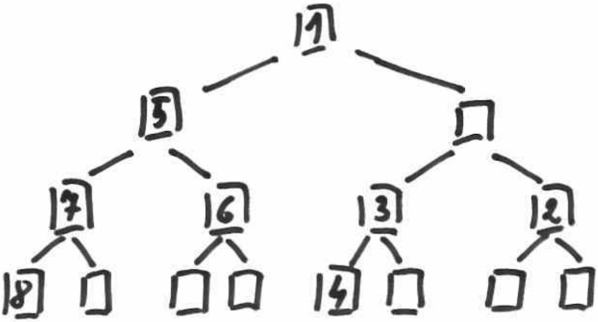
1)



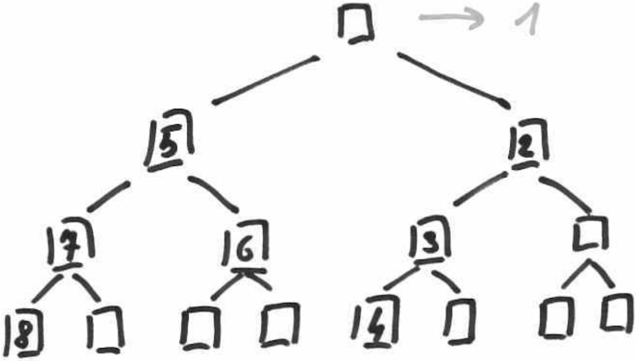
2)



3)



4)



atd.

<u>Složitost:</u>

$P(n) = 2n-1 = O(n)$

čas: 1. minimum se dostane do kořene
v $\log n$ krocích
v 1 kroku se odebere

2. a další minimum se najde v 1 kroku
v 1 kroku se odebere

$T(n) = \log n + 1 + 2(n-1) = 2n + \log n - 1 = O(n)$

cena: $C(n) = O(n^2)$

není optimální

Bucketsorting and merging

předpoklady: $n = 2^m$

m je mocnina 2

strom má m listů

$\log m + 1$ hladin

$2m - 1$ procesorů

1) každý list obsahuje $\frac{n}{m}$ prvků

obsah se setřídí (Heapsortem)

2) každý nelistový procesor zmerguje

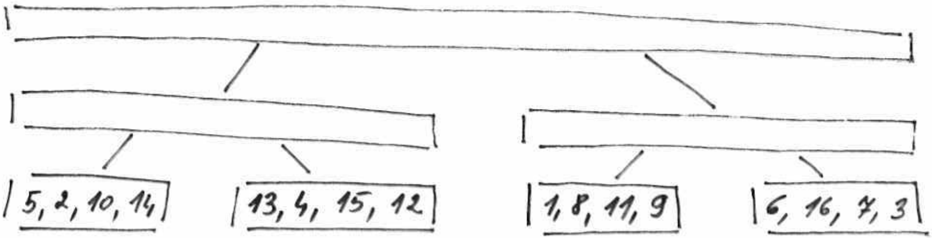
obsah svých synů

3) výsledek je v kořeni

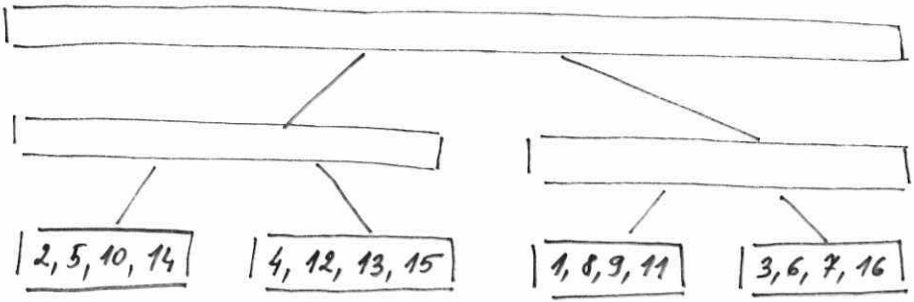
Příklad: třídím 5, 2, 10, 14, 13, 4, 15, 12, 1, 8, 11, 9, 6, 16, 7, 3

$m = 4$

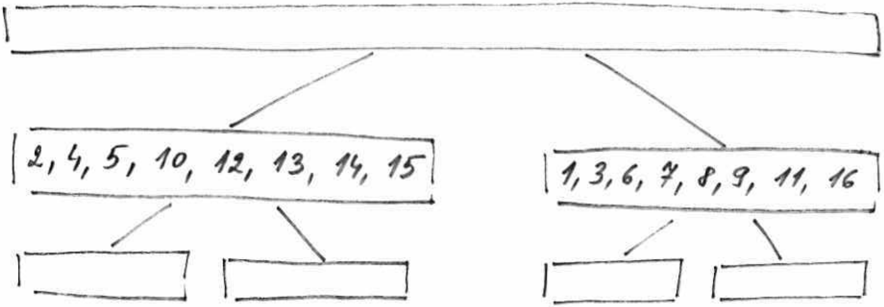
0)



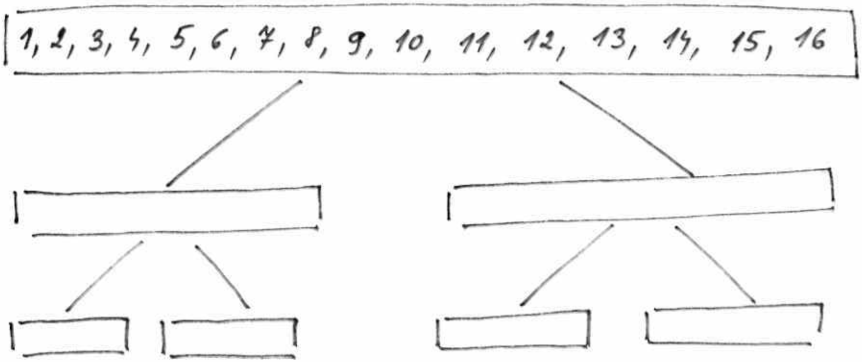
1)



2)



3)



<u>Složitost:</u>

$P(n) = 2m - 1 = 2\log n - 1 = O(\log n)$

čas: čtení vstupních hodnot $O(\frac{n}{\log n})$

Heapsort $O(\frac{n}{\log n} \log \frac{n}{\log n}) = O(n)$

mergování: procesor na hladině i merguje

2 posloupnosti délky $\frac{n}{2^{i+1}}$ do jedné

posloupnosti délky $\frac{n}{2^i}$

čas: $\frac{n}{2^i}$

celkem: $\sum_{i=0}^{\log m - 1} \frac{n}{2^i} = n \sum_{i=0}^{\log m - 1} \left(\frac{1}{2}\right)^i \leq 2n = O(n)$

cena: $C(n) = O(n \log n)$

je optimální

Poznámky:

1) Dá se třídit více posloupností najednou – další třídění může začít, jakmile se uvolní listové procesory.

2) Vnitřní procesory nemusí mít tak velkou kapacitu – stačí 2 prvky.
(mergování může začít, jakmile jsou dostupné první prvky z každé podposloupnosti)

Median-splitting sort

stejný strom jako v předchozím případě

postup opačný:

vstupní data jsou v kořeni

najde se medián m

prvky < m se pošlou do levého syna

prvky > m -"- pravého -"-

atd.

v listech se použije Heapsort

Složitost:

stejná

3) Mnoho dalších architektur

SIMD SE SDÍLENOU PAMĚTÍ

paralelní RAM EREW

Algoritmus se má přizpůsobit počtu procesorů, které jsou k dispozici

n ... délka tříděné posloupnosti

N ... počet procesorů, $N < n$

vyjádříme $N = n^{1-c}$, c je konstanta mezi 0 a 1

Algoritmus má být cenově optimální, tj. $C(n) = O(n \log n)$

musíme dosáhnout čas $O(n^c \log n)$

Pomocné procedury:

Broadcast - čtení téže hodnoty více procesory

Parallel select - paralelní výpočet mediánu (k-tého nejmenšího prvku)

Výsledný algoritmus:

Sharesort - paralelní verze Quicksortu

AKL 1984

Broadcast

používá pomocné pole $B(1), \dots, B(N)$ ve sdílené paměti

procesor P_1 přečte hodnotu m a zapíše ji do $B(1)$

P_2 — " — $B(1)$ — " — $B(2)$

P_3, P_4 současně čtou $B(1), B(2)$ a zapisují do $B(3), B(4)$

$P_5, \dots, P_8$ — " — $B(1), \dots, B(4)$ — " — $B(5), \dots, B(8)$

atd.

procedure BROADCAST (m, N, B)

P_1 přečte m a zapíše m do $B(1)$

for $i := 0$ **to** $\log N - 1$ **do**

for $j := 2^i + 1$ **to** 2^{i+1} **do in parallel**

P_j přečte $B(j - 2^i)$ a zapíše tuto hodnotu do $B(j)$

enddo

enddo

čas: $O(\log N)$

Paralelní výpočet k-tého nejmenšího prvku

máme procesory $P_1, \dots, P_{n^{1-e}}$

hodnoty n, k a pole S jsou ve sdílené paměti

hodnota e je procesorům známá

procedure PARALLEL SELECT (S, k)

if $|S| < 3$ **then** provedeme 1 porovnání na 1 procesoru konec

else rozdělíme S na $|S|^{1-e}$ částí po $|S|^e$ prvcích

každá část se přiřadí jednomu procesoru

endif

for $i := 1$ **to** $|S|^{1-e}$ **do in parallel**

P_i najde medián svého úseku (sekvenčním SELECTem)

zapíše jeho hodnotu m_i do $M(i)$ ve sdílené paměti

enddo

PARALLEL SELECT $(M, \lceil \frac{|M|}{2} \rceil)$ - vrátí hodnotu m, která je mediánem M

vytvoří se množiny S_1, S_2, S_3 prvků S menších, rovných a větších než m

<u>if</u> $|S_1| \geq k$ <u>then</u> PARALLEL SELECT(S_1, k)

<u>else</u> <u>if</u> $|S_1| + |S_2| \geq k$ <u>then</u> vrátí se hodnota m

<u>else</u> PARALLEL SELECT$(S_3, k - |S_1| - |S_2|)$

<u>endif</u>

<u>endif</u>

Implementace a analýza

1) čtení vstupních dat:

počáteční adresa A pole S, hodnoty n, k se dostanou k procesorům Broadcastem

čas: $O(\log n^{1-\epsilon})$

2) rozdělení pole S procesorům:

každý P_i spočítá adresu prvního a posledního prvku svého úseku: $A + (i-1)n^c$

$A + in^c - 1$

čas: $O(1)$

3) výpočet mediánu každého úseku:

čas: $O(n^c)$

4) výpočet mediánu M:

čas: $t(n^{1-c})$

5) rozdělení S na S_1, S_2, S_3:

a) hodnota m se Broadcastem pošle všem procesorům

čas: $O(\log n^{1-c})$

b) každý P_i rozdělí svůj úsek podle m na S_1^i, S_2^i, S_3^i

čas: $O(n^c)$

c) vytvoří se S_1 spojením všech S_1^i takto:

nechť $s_i = |S_1^i|$

pro každé i se vypočte $z_i = \sum_{j=1}^{i} s_j$

procesory současně zapíší své S_1^i do S_1 tak, že

P_i začíná zapisovat od pozice $z_{i-1}+1$

(platí $z_0 = 0$, $z_{n^{1-c}} = |S_1|$)

čas zápisu: $O(n^c)$

paralelní výpočet z_i:

<u>procedure</u> ALLSUMS

<u>for</u> $j := 0$ <u>to</u> $\log N - 1$ <u>do</u>

<u>for</u> $i := 2^j + 1$ <u>to</u> N <u>do</u> <u>in parallel</u>

P_i přečte s_{i-2^j}

vypočte a zapíše $s_i := s_i + s_{i-2^j}$

<u>enddo</u>

<u>enddo</u>

čas: $O(\log N)$

Příklad:

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8
$j=0$		$+S_1$	$+S_2$	$+S_3$	$+S_4$	$+S_5$	$+S_6$	$+S_7$
$j=1$			$+S_1$	$+S_2$	$+S_3$	$+S_4$	$+S_5$	$+S_6$
				$+S_1$	$+S_2$	$+S_3$	$+S_4$	$+S_5$
$j=2$					$+S_1$	$+S_2$	$+S_3$	$+S_4$
						$+S_1$	$+S_2$	$+S_3$
							$+S_1$	$+S_2$
								$+S_1$

d) vytvoří se S_2, S_3 - analogicky

e) PARALLEL SELECT (S_1, k)

-"- $(S_3, k-|S_1|-|S_2|)$

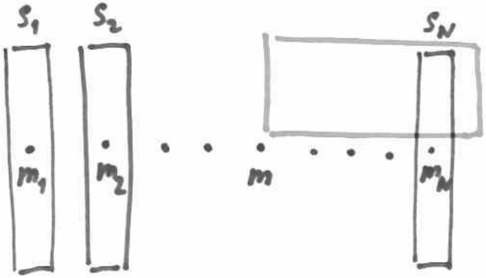
čas: nejvýše $t\left(\frac{3n}{4}\right)$

Důkaz:

m je medián M, $|M| = n^{1-\varepsilon}$ $\Rightarrow$

$\frac{n^{1-\varepsilon}}{2}$ prvků S je větších než m

každý prvek M je menší než alespoň $\frac{n^{\varepsilon}}{2}$ prvků S



prvky větší než m
je jich:
$\frac{n^{1-\varepsilon}}{2} \cdot \frac{n^{\varepsilon}}{2} = \frac{n}{4}$
nebo více

zbylé prvky jsou menší než m

je jich $\frac{3n}{4}$ nebo méně

z toho: $|S_1| \leq \frac{3n}{4}$

analogicky také $|S_3| \leq \frac{3n}{4}$

<u>čas algoritmu celkem:</u>

$$t(n) = O(\log n^{1-c}) + O(n^c) + t(n^{1-c}) + t\left(\frac{3n}{4}\right)$$

řešení: $t(n) = O(n^c)$

důkaz dosazením:

$$\underbrace{\log n^{1-c}}_{< \log n < n^c} + n^c + \underbrace{(n^{1-c})^c}_{< n^c} + \left(\frac{3n}{4}\right)^c = O(n^c)$$

<u>cena</u>: $O(n^{1-c} n^c) = O(n)$

je to optimální algoritmus

Sharesort

vychází ze sekvenčního Quicksortu:

procedure Quicksort

  if $|S| < 3$ **then** provedeme 1 porovnání a konec

    else vypočteme m = medián S

  endif

  S_1 := prvky S menší než m

  S_2 := — " — větší — " —

  umístíme m na pozici $\lceil \frac{|S|}{2} \rceil$

  Quicksort(S_1)

  Quicksort(S_2)

složitost: $O(n \log n)$

1. idea paralelizace (špatná):

provést Quicksort(S_1) a Quicksort(S_2) současně

k tomu nemáme dost procesorů

problém velikosti n:

k dosažení času $t(n)$ potřebuji n^{1-c} procesorů

problém velikosti $\frac{n}{2}$:

k dosažení času $t(\frac{n}{2})$ potřebuji $(\frac{n}{2})^{1-c}$ procesorů

ale na každou polovinu jich můžu dát jen $\frac{n^{1-c}}{2}$

2. idea (dobrá):

rozdělit S na víc částí podle více pivotů

pro $i = 1, 2, \ldots, 2^{1/c} - 1$ najdu pivoty m_i tak, že

$m_i = \lceil \frac{in}{2^{1/c}} \rceil$-tý nejmenší prvek S

Příklad: pro $c = 0{,}5$ je $2^{1/c} - 1 = 3$

hledám m_1, m_2, m_3

$m_1 = \frac{n}{4}$-tý nejmenší

$m_2 = \frac{n}{2}$-tý nejmenší (medián)

$m_3 = \frac{3n}{4}$-tý nejmenší

podle m_i se S rozdělí na $2^{1/k}$ podposloupností

každá má délku $\frac{n}{2^{1/k}}$

označení:

$$\underbrace{S_1^1, S_1^2, \ldots, S_1^{2^{1/k-1}}}_{S_1}, \underbrace{S_2^1, S_2^2, \ldots, S_2^{2^{1/k-1}}}_{S_2}$$

algoritmus se aplikuje paralelně v každé polovině

napřed v S_1, potom v S_2

použije se $\frac{n^{1-\varepsilon}}{2^{1/k-1}}$ procesorů na podposloupnost

$$\frac{n^{1-\varepsilon}}{2^{1/k-1}} = \left(\frac{n}{2^{1/k}}\right)^{1-\varepsilon}$$ dostatečný počet pro rekurzi

Algoritmus pro $c = 0{,}5$

procedure Sharesort (S)

if $|S| < 3$ **then** 1 porovnání a konec

else najdeme m_1, m_2, m_3

endif

rozdělíme S: $S_1^1 = \{x_i \in S : x_i < m_1\}$

$S_1^2 = \{x_i \in S : m_1 < x_i < m_2\}$

$S_2^1 = \{x_i \in S : m_2 < x_i < m_3\}$

$S_2^2 = \{x_i \in S : m_3 < x_i\}$

m_1, m_2, m_3 dáme na pozice $\lceil \frac{|S|}{4} \rceil, \lceil \frac{|S|}{2} \rceil, \lceil \frac{3|S|}{4} \rceil$

do in parallel

Sharesort (S_1^1)

Sharesort (S_1^2)

enddo

do in parallel

Sharesort (S_2^1)

Sharesort (S_2^2)

enddo

Složitost

nalezení m_i a rozdělení do podposl. S_1^i, S_2^i

použijeme Parallel select

čas: $O(n^c)$

rekurze pro Sharesort:

$$t(n) = konst. \, n^c + 2t\left(\frac{n}{2^{1/c}}\right)$$

řešení: $t(n) = O(n^c \log n)$

počet procesorů: n^{1-c}

cena: $O(n \log n)$

je to optimální algoritmus

Poznámka:

Existují i jiné verze paralelního Quicksortu s cenou $O(n \log n)$ v průměrném případě, $O(n^2)$ v nejhorším případě

MIMD – ASYNCHRONNÍ TŘÍDĚNÍ

algoritmus = soubor procesů

někteří se provádějí současně na procesorech, které jsou momentálně k dispozici

není-li žádný volný procesor, proces čeká ve frontě
–"– čekající proces, procesor –"–

přiřazování procesů procesorům – existují různé způsoby

režim fronty – různé způsoby

Enumeration sort

pro každý prvek se spočítá, kolik je v posloupnosti prvků menších než on

to určí jeho výslednou pozici

sekvenční algoritmus – má složitost $O(n^2)$

paralelní asynchronní algoritmus:

pro každý prvek x_i se vytvoří proces,
který ho porovná se všemi ostatními
a zařadí na správnou pozici

každý proces pracuje nezávisle na ostatních
(nemusí s nimi komunikovat)

<u>Algoritmus</u> (X je původní pole, T je výsledné pole):

```
for i:= 1 to n do vytvoř proces i enddo
proces i:
  k:=0
  for j:=1 to n do
     if X(i) > X(j) then k:=k+1
          else if X(i) = X(j) and i>j then k:=k+1
               endif
     endif
  enddo
  T(k+1) := X(i)
```

předpokládaný přístup ke sdílené paměti:

CREW

Složitost

n procesů

každý vykoná $O(n)$ operací

máme-li p procesorů, pak:

paralelní čas $T(n) = \frac{n}{p} O(n) = O(\frac{n^2}{p})$

cena $c(n) = p\, O(\frac{n^2}{p}) = O(n^2)$

Příklad: třídím 8, 6, 6, 9, 7

mám procesory P_1, P_2

P_1 vytvoří 5 procesů

procesy 1 a 2 (tj. pro 8 a 6) běží paralelně na P_1 a P_2

proces 1 skončí dřív (tam je skoro vždy $X(i) > X(j)$ a rovnost se netestuje)

P_1 se uvolní dřív a začne zpracovávat proces 3 proces 4 poběží na P_2

se zpožděním oproti procesu 3

je ale kratší, může doběhnout dřív

proces 5 pak poběží na P_2

Kvůli asynchronnímu zpracování je problém předem odhadnout chování programu.

LITERATURA

např. S. G. Akl: Parallel sorting algorithms (Academic Press, 1985)

existuje i novější monografie téhož autora