Paralelné výpočty v konštantnom čase

PRAM model

- **PRAM** = **Parallel Random Access Machine**
  - model s (neobmedzenou) zdieľanou pamäťou
  - **synchrónny model** – spoločné hodiny
  - **SIMD** – v každom kroku sa vykonáva rovnaká inštrukcia
  - vstup aj výstup v zdieľanej pamäti
  - každý procesor má ID a pozná celkový počet procesorov
- Prístup do spoločnej pamäte:
  - **global_read**(Shared, Local)
  - **global_write**(Local, Shared)

PRAM – prístup do pamäte

- rôzne varianty pri súčasnom prístupe na rovnaké miesto zdieľanej pamäte:

  EREW PRAM – Exclusive Read, Exclusive Write

  CREW PRAM – Concurrent Read, Exclusive Write

  CRCW PRAM – Concurrent Read, Concurrent Write

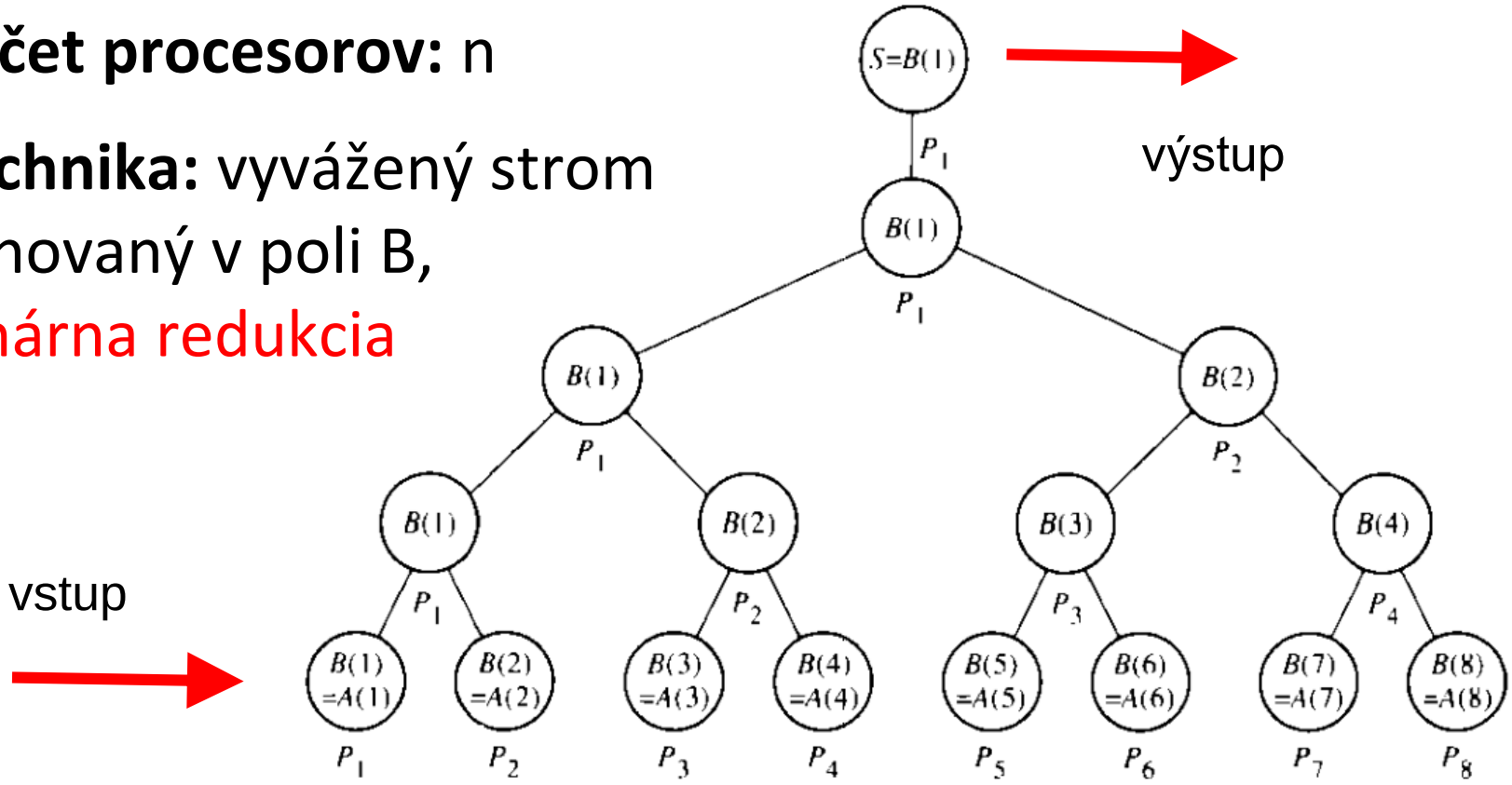
  - Common – pri zápise sa zhodujú hodnoty
  - Arbitrary – uspeje ľubovoľný
  - Priority – uspeje procesor s najnižším indexom

Výkon paralelných algoritmov

- $T^*(n)$ – sekvenčný čas – počet krokov optimálneho sekvenčného algoritmu
- $T_p(n)$ – paralelný čas pri použití p procesorov
- Span (hranice možnosti paralelizácie) $T_\infty(n)$
- Zrýchlenie (speed-up) $S_p(n) = T^*(n) / T_p(n)$
- Cena (cost) $C_p(n) = p.T_p(n)$
- Cenovo optimálny algoritmus $C_p(n) \in O(T^*(n))$
- Efektívnosť $E_p(n) = T^*(n)/C_p(n) = S_p(n)/p$

Paralelné sčítanie (binárna redukcia)

- **Vstup:** pole A[1..n], kde n = 2^k
- **Výstup:** súčet prvok poľa A v premennej S
- **Počet procesorov:** n
- **Technika:** vyvážený strom uchovaný v poli B, binárna redukcia



Paralelné sčítanie (binárna redukcia)

Kód pre procesor i ($1 \leq i \leq n$ ... stačí $n/2$):

```
global_read(A[i], x);
for h:=1 to log(n) do
   if (i <= n/2^h) then
   begin
     global_read(A[n/2^h + i], y);
     x := x + y;
     global_write(x, A[i]);
   end;
if i = 1 then global_write(x, S);
```

pre $n \neq 2^k$?

EREW model

$T_n(n) \in \Theta(\log(n))$

$S_n(n) \in \Theta(n/\log(n))$

$C_n(n) \in \Theta(n.\log(n))$

$E_n(n) \in \Theta(1/\log(n))$

nie je cenovo optimálny

Prefixové súčty

- **Vstup:**
  - binárna asociatívna operácia + (*, min, max, or, …)
  - n-prvkové pole A, kde n = 2^k
- **Výstup**:
  - n-prvkové pole S také, že S[i] = A[1] + A[2] + … + A[i]
- **Sekvenčný algoritmus v čase O(n):**

```
S[1] := A[1];
for i := 2 to N do
  S[i] := S[i-1] + A[i];
```

Hillis-Steele prefix sum

Hillis-Steelov algoritmus na výpočet prefixových súčtov
kód pre procesor i ($1 \leq i \leq n$):

```
global_read(A[i], x);
global_write(x, S[i]);
for h := 0 to log(n)-1 do
   if i > 2^h then
   begin
       global_read(S[i-2^h], y);
       x := x + y;
       global_write(x, S[i]);
   end;
```

EREW model

$T^*(n) = n$

$T_n(n) \in \Theta(\log(n))$

$S_n(n) \in \Theta(n/\log(n))$

$C_n(n) \in \Theta(n.\log(n))$

$E_n(n) \in \Theta(1/\log(n))$

Paralelné vyhľadávanie v konštantnom čase

- n procesorov P_1, P_2, …, P_n
- **Vstup**:
  - pole M[1..n] - obsahujúce zoznam n čísel
  - M[n+1] – hľadaná hodnota
- **Výstup**:
  - M[n+2] - index prvku poľa, ktoré obsahuje M[n+1]
- Kód pre procesor P_i:

```
global_read(M[n+1], x);                          CR
global_read(M[i], y);                            ER
if i=1 then global_write(0, M[n+2]);             EW
if x=y then global_write(i, M[n+2]);             CW-arbitrary
```

CRCW-arbitrary

Paralelné vyhľadávanie v EREW PRAM

Kód pre procesor i ($1 \leq i \leq n$):

```
for h:=1 to log(n) do
   if (i <= 2^(h-1)) then
   begin
   global_read(M[n+i], x);
   global_write(x, M[n+i+2^(h-1)]);
   end;
global_read(M[n+i], x);
global_read(M[i], y);
if x = y
  then global_write(i, M[n+i])
  else global_write(0, M[n+i]);
```

```
for h:=1 to log(n) do
   if (i <= n/2^h) then
   begin
     global_read(M[n+2*i-1], x);
     global_read(M[n+2*i], y);
     if x < y then z := y
              else z := x;
     global_write(z, M[n+i]);
   end;
if i = 1 then
    global_write(z, M[n+2]);
```

hľadanie maximálneho prvku

- nájsť maximum hodnôt z poľa A[1..n]
  - binárny strom v čase *O(log(n))* (ako súčet)

```
global_read(A[i], x);
for h:=1 to log(n) do
   if (i <= n/2^h) then
   begin
     global_read(A[n/2^h + i], y);
     if y > x then x := y;
     global_write(x, A[i]);
   end;
if i = 1 then global_write(x, Maximum);
```

hľadanie maxima v konštantnom čase

čiastočný problém

- je A[j] maximálny ? (porovnať hodnotu A[j] s ostatnými)

```
global_write(1, M[j]);     { CW-common }
global_read(A[j], x);      { CR }
global_read(A[i], y);      { ER }
if x < y then global_write(0, M[j]);   { CW-common }
```

{ na 4 kroky v M[j] ostane 1 práve ak je A[j] maximálny ... }

- paralelné riešenie pre každý prvok poľa A ?
  potrebujeme n*n procesorov

maximum v čase O(1) s n^2 procesormi

pre hodnoty A[0..n-1] nájsť maximum
n*n procesorov (0 … n^2-1)

```
if i <= (n-1) then
        global_write(1, M[i]);
global_read(A[i div n], x);
global_read(A[i mod n], y);
if x < y then
        global_write(0, M[i div n]);
y := 0;
if i <= (n-1) then global_read(M[i], y);
if y = 1 then global_write(x, Maximum);
```

rýchly, ale neefektívny

CRCW - common
1 ostane len pre to i, pre ktoré je vždy A[i] >= A[j]

$T^*(n) = n$

$T_{n*n}(n) \in \Theta(1)$

$S_{n*n}(n) \in \Theta(n)$

$C_{n*n}(n) \in \Theta(n^2)$

$E_{n*n}(n) \in \Theta(1/n)$

maximum s menšou cenou

lepšie využitie procesorov (odmocninová redukcia) :

- rozdeliť úlohu na √n častí – v každej je pole veľkosti √n
- v 1. kole rieši každú úlohu √n * √n = n procesorov celkove √n * n procesorov, každý v konštantnom čase
- v 2. kole už je len jedna úloha pre pole veľkosti √n, ktorú vyrieši n procesorov v konštantnom čase
- celkove $T(n) \in O(1)$, $C(n) \in O(n.\sqrt{n})$, $E(n) \in O(1/\sqrt{n})$
- model CRCW-common
- je možné ešte ďalšie zlepšenie ?

maximum v čase O(1) s $n^{1,5}$ procesormi

pre hodnoty A[0..n-1] nájsť maximum

$n=m^2$ $n*\sqrt{n}=m^3$ procesorov v 3D poli (i,j,k) $0 \le i,j,k \le m-1$

```
global_write(1, P[i,j]);
global_read(A[k*m + i], x);
global_read(A[k*m + j], y);
if x < y then global_write(0, P[k,i]);
global_read(P[k,i], y);
if y = 1 then global_write(x, B[k]);
global_write(1, M[i]);
global_read(B[i], x);  global_read(B[j], y);
if x < y then global_write(0, M[i]);
global_read(M[i], y);
if y = 1 then Maximum := A[i];
```

pomocné pole P m*m indikátorov pre každú časť veľkosti m

pole B veľkosti m pre maximá jednotlivých častí

CRCW - common

$T^*(n) = n$

$T_{n/\sqrt{n}}(n) \in \Theta(1)$

$S_{n/\sqrt{n}}(n) \in \Theta(n)$

$C_{n/\sqrt{n}}(n) \in \Theta(n\sqrt{n})$

$E_{n/\sqrt{n}}(n) \in \Theta(1/\sqrt{n})$

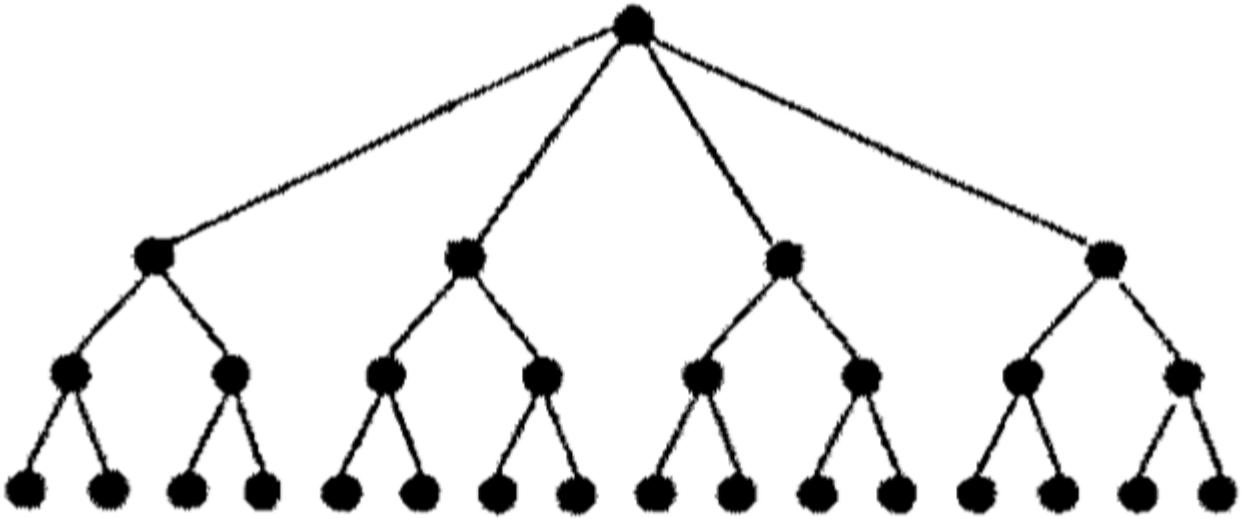
maximum s ešte menšou cenou

- rozdeliť úlohu na √n častí – v každej je pole veľkosti √n
- každú úlohu pre pole veľkosti √n rozdeliť na √√n častí – v každej je pole veľkosti √√n
- v 1. kole rieši každú úlohu veľkosti √√n √n procesorov v konštantnom čase – celkom √n.√√n .√n = n . √√n procesorov
- v 2. kole prebieha riešenie v √n častiach, ale každá už len s √√n prvkami – stačí √n * √n = n procesorov
- v 3. kole už je len jedna úloha pre pole veľkosti √n, ktorú vyrieši opäť n procesorov v konštantnom čase
- celkove $T(n) \in O(1)$, $C(n) \in O(n.\sqrt{\sqrt{n}}) = O(n^{1,25})$, $E(n) \in O(1/n^{¼})$
- čo tak rozdeliť pole veľkosti √√n ďalej na √√√n častí ?

dvojlogaritmický strom

- uzol, z ktorého je dostupných x listov rozvetvíme na √x nasledovníkov
- pre $n = 2^{2^k}$ to znamená
  - koreň má $\sqrt{n} = 2^{2^{k-1}}$ nasledovníkov
  - uzol na i-tej úrovni (i=0 je koreň) má $2^{2^{k-i-1}}$ nasledovníkov
  - na úrovni 1 je $2^{2^{k-1}}$ uzlov
    na úrovni 2 je $2^{2^{k-1}} . 2^{2^{k-2}} = 2^{2^{k-1}+2^{k-2}} = 2^{2^k-2^{k-2}}$ uzlov
    …
    na úrovni i je $2^{2^k-2^{k-i+1}} . 2^{2^{k-i}} = 2^{2^k-2^{k-i+1}+2^{k-i}} = 2^{2^k-2^{k-i}}$ uzlov
  - na úrovni k bude $2^{2^k-2^{k-k}} = n/2$ uzlov, každý s 2 listami
  - výška stromu je $\log(\log(n))+1$

maximum v čase O(loglog(n)) s n procesormi



v uzle s *x* nasledovníkmi vieme spočítať maximum z ich hodnôt v čase O(1) pomocou x^2 procesorov

na úrovni i je $2^{2^k-2^{k-i}}$ uzlov, každý má $2^{2^{k-i-1}}$ nasledovníkov, potrebujeme $2^{2^k-2^{k-i}}.(2^{2^{k-i-1}})^2 = 2^{2^k}$ = n procesorov

$T(n) \in O(\log\log(n))$, $C(n) \in O(n.\log\log(n))$, $E(n) \in O(1/\log\log(n))$

Amdahlov zákon

- Sekvenčný program s trvaním T(n):
  - neparalelizovateľná časť: $f_s.T(n)$
  - paralelizovateľná časť : $(1-f_s).T(n)$
- Ohraničenie zrýchlenia programu (v ideálnom prípade) pri p procesoroch:
  - $T_p(n) = f_s.T(n) + (1-f_s).T(n)/p$
  - $S_p(n) = T(n)/ T_p(n)=1/(f_s +(1-f_s)/p)$
- $S_\infty(n) = 1/ f_s$
- pridávaním procesorov nemožno zväčšovať zrýchlenie do nekonečna

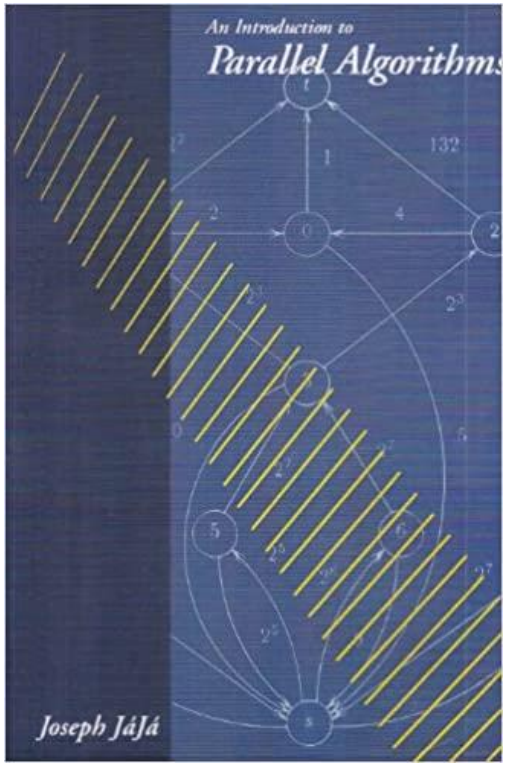
Gustafsonov zákon

- Paralelný program pre p procesorov s trvaním $T_p(n)$
  - neparalelizovaná časť: $g_s.T_p(n)$
  - paralelizovaná časť : $(1-g_s).T_p(n)$
- Sekvenčný čas: $T(n) = g_s.T_p(n) + p.\ (1-g_s).T_p(n)$
- Zrýchlenie:
  - $S_p(n) = T(n)/\ T_p(n)=g_s + p.\ (1-g_s) = p.\ (1-g_s-g_s/p) \approx p.\ (1-g_s)$
  - ak úlohu necháme v tom istom čase riešiť viacerými procesormi (s väčším objemom vstupu – teda zväčšovaním podielu paralelnej zložky na celom výpočte), zrýchlenie vzrastie lineárne

študijná literatúra

- Joseph Jájá - An Introduction to Parallel Algorithms

ISBN-13: 978-0201548563
ISBN-10: 0201548569



Ďakujem za pozornosť !