Projektovanje paralelnih algoritama III

- Primeri paralelnih algoritama, II deo
  - Dve verzije paralelnog algoritma za sortiranje sa spajanjem podnizova

I verzija paralelnog algoritma sortiranja sa spajanjem podnizova

- Za serijski algoritam $T(n) = \Theta(n \lg n)$
- Serijski algoritm koristi pristup podeli-i-zavladaj
  - Paralelizacija korišćenjem ugnježdenog paralelizma
  - Prvi rekurzivni poziv postaje izmrešćen poziv:

```
Merge-Sort-P(A, p, r)
1.if p < r
2.    q = ⌊(p+r)/2⌋
3.    spawn Merge-Sort-P(A, p, q)
4.    Merge-Sort-P(A, q+1, r)
5.    sync
6.    Merge(A, p, q, r)
```

- Kao njegov serijski dvojnik, Merge-Sort-P sortira podniz $A[p..r]$
- Procedura čeka završetak dva rekurzivna poziva putem iskaza **sync** u liniji 5
- Iza toga poziva proceduru Merge koju koristi i serijski dvojnik

Analiza procedure Merge-Sort-P (1/2)

- Rad:
  - Pošto je procedura Merge serijska, njen rad i raspon su oba $\Theta(n)$
  - Rekurencija za rad $T_1(n)$ procedure Merge-Sort-P:

    $T_1(n) = 2\ T_1(n/2) + \Theta(n) = \Theta(n \lg n)$
  - Rad je dakle isti kao $T(n)$ serijskog dvojnika
- Raspon:
  - Kako se dva rekurzivna poziva mogu izvršavati u ||:

    $T_\infty(n) = T_\infty(n/2) + \Theta(n) = \Theta(n)$
- Paralelizam: $T_1(n)/T_\infty(n) = \Theta(\lg n)$

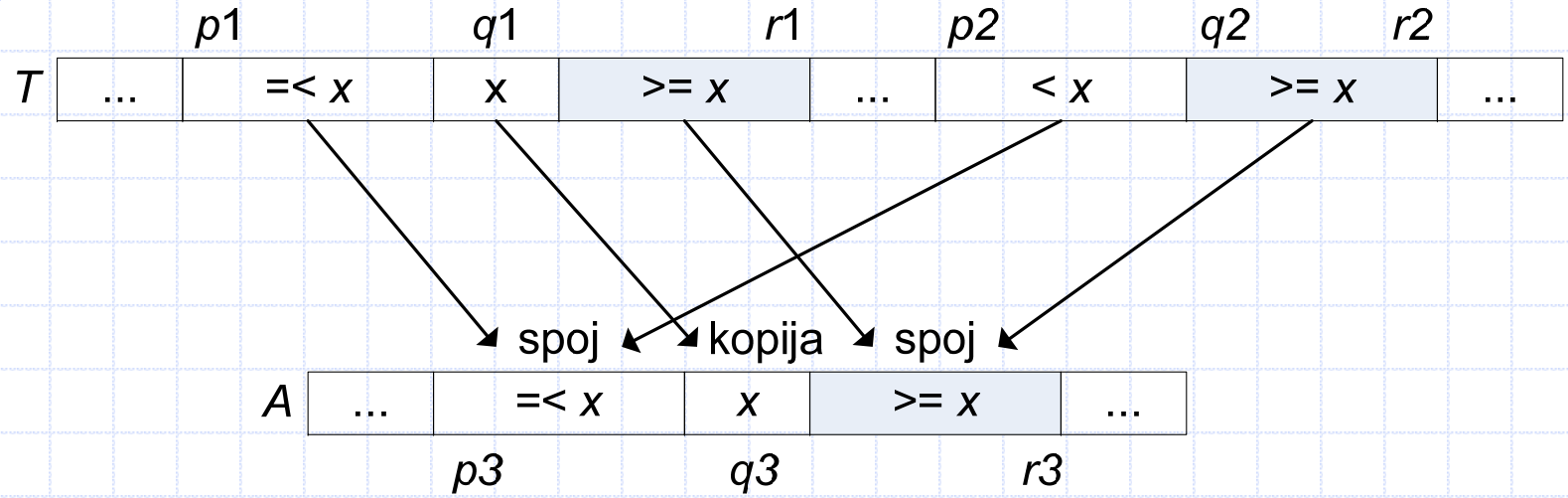
Analiza procedure Merge-Sort-P (2/2)

- Ova procedura može da ostvari linearno ubrzanje na nekoliko procesora
  - Ali ne može se efektivno skalirati na 100-ne procesora
- Gde je usko grlo paralelizma?
  - To je serijska procedura Merge
- Prirodno serijska radnja? Da li se može ||?
  - Da, spajanje je moguće paralelizovati korišćenjem ugnježdenog paralelizma

Strategija podeli-i-zavladaj za paralelno spajanje podnizova (1/3)

- Spaja dva sortirana podniza:
  - $T[p_1..r_1]$ dužine $n_1 = r_1 - p_1 + 1$ i
  - $T[p_2..r_2]$ dužine $n_2 = r_2 - p_2 + 1$
  - u drugi podniz $A[p_3..r_3]$ dužine $n_3 = r_3 - p_3 + 1$
- Uprošćavajuća pretpostavka: $n_1 \geq n_2$
- Dva ključna koraka na početku:
  - Pronaći srednji element $x = T[q_1]$, podniza $T[p_1..r_1]$, gde je $q_1 = \lfloor (r_1 + p_1)/2 \rfloor$; q_1 je median!
  - Pronaći indeks q_2 niza $T[p_2..r_2]$ takav da podniz ostaje sortiran nakon umetanja x između $T[q_2 - 1]$ i $T[q_2]$.

Strategija podeli-i-zavladaj za paralelno spajanje podnizova (2/3)



- Nakon održivanja q_1 i q_2, sledi samo spajanje:

1. Postavi $q_3 = p_3 + (q_1 - p_1) + (q_2 - p_2)$
2. Kopiraj x u $A[q_3]$
3. Rekurzivno spajaj $T[p_1..q_1\text{-}1]$ i $T[p_2..q_2\text{-}1]$ i smesti rezultat u podniz $A[p_3..q_3\text{-}1]$
4. Rekurzivno spajaj $T[q_1\text{+}1..r_1]$ i $T[q_2..r_2]$ i rezultat smesti u podniz $A[q_3\text{+}1..r_3]$

Strategija podeli-i-zavladaj za paralelno spajanje podnizova (3/3)

- Prilikom računanja q_3 vrednost:
  - $q_1 - p_1$ je broj prethodnih elem. iz podniza $T[p_1..q_1\text{-}1]$
  - $q_2 - p_2$ je broj prethodnih elem. iz podniza $T[p_2..q_2\text{-}1]$
  - Njihova suma je broj elemenata koji prethode elementu x u podnizu $A[p_3..r_3]$
- Osnovni slučaj: za $n_1 = n_2 = 0 \Rightarrow$ nema posla
- Kako je $n_1 \geq n_2$, osnovni slučaj: $n_1 = 0$
- Ako je samo jedan od podnizova prazan, to mora biti podniz $T[p_2..r_2]$ (jer je $n_1 \geq n_2$)

Procedura Binary-Search(x, T, p, r)

```
Binary-Search(x, T, p, r)
1.low = p
2.high = max(p, r + 1)
3.while low < high
4.    mid = ⌊(low+high)/2⌋
5.    if x ≤ T[mid]
6.        high = mid
7.    else low = mid + 1
8.return high
```

- Uzima ključ x i podniz $T[p..r]$ i vraća jedan od sledeća tri rezultata:
  1. Ako je $T[p..r]$ prazan ($r < p$), onda vraća indeks p
  2. Ako je $x \leq T[p]$, pa time manji ili jednak od svih elemenata $T[p..r]$, onda vraća indeks p
  3. Ako je $x > T[p]$, onda vraća najveći indeks q, u opsegu $p < q \leq r + 1$, takav da je $T[q-1] < x$

Analiza Procedure Binary-Search

- Poziv procedure uzima $\Theta(\lg n)$ serijskog vremena u najgorem slučaju
  - gde je $n = r - p + 1$ veličina podniza na kom se procedura izvršava
- Pošto je Binary-Search serijska procedura, njen rad i raspon su u najgorem slučaju $\Theta(\lg n)$, oba

Procedura P-Merge (paralelno spajanje ponizova)

P-Merge(T, $p1$, $r1$, $p2$, $r2$, A, p_3)
1. $n_1 = r_1 - p_1 + 1$
2. $n_2 = r_2 - p_2 + 1$
3. **if** $n_1 < n_2$ // osiguraj da je $n_1 \geq n_2$
4. zameni p_1 sa p_2
5. zameni r_1 sa r_2
6. zameni n_1 sa n_2
7. **if** $n_1 == 0$ // oba podniza prazna?
8. **return**
9. **else** $q_1 = \lfloor (p_1 + r_1)/2 \rfloor$
10. q_2 = Binary-Search($T[q_1]$, T, p_2, r_2)
11. $q_3 = p_3 + (q_1 - p_1) + (q_2 - p_2)$
12. $A[q_3] = T[q_1]$
13. **spawn** P-Merge(T, p_1, $q_1 - 1$, p_2, $q_2 - 1$, A, p_3)
14. P-Merge(T, $q_1 + 1$, r_1, q_2, r_2, A, $q_3 + 1$)
15. **sync**

- Spaja podnizove $T[p_1..r_1]$ i $T[p_2..r_2]$ u podniz $A[p_3..r_3]$
- 3-6: $n_1 \geq n_2$
- 7: osnovni slučaj?
- 9-15: realizuju strategiju podeli-i-zavladaj
- 10-11: q_2 i q_3
- 12: direktno kopira $T[q_1]$ u $A[q_3]$
- 13-15: ugnježdeni paralelizam

Analiza procedure P-Merge (1/4)

- Raspon:
  - Dva podniza sadrže ukupno $n = n_1 + n_2$ elemenata
  - Rekurzivni pozivi rade ||. Koji je skuplji?
  - Ključ: max br. elemenata u bilo koja od ta dva rekurzivna poziva može biti najviše $3n/4$
    - Kako je $n_2 \leq n_1$, sledi: $n_2 = 2n_2/2 \leq (n_1+n_2)/2 = n/2$
    - U najgorem slučaju, jedan od dva rekurzivna poziva spaja $\lfloor n_1/2 \rfloor$ elemenata podniza $T[p_1..r_1]$ sa svih n_2 elemenata podniza $T[p_2..r_2]$, tj. ukupno spaja ovoliko elemenata:
      $\lfloor n_1/2 \rfloor + n_2 \leq n_1/2 + n_2/2 + n_2/2 = (n_1 + n_2)/2 + n_2/2$
      $\leq n/2 + n/4 = 3n/4$

Analiza procedure P-Merge (2/4)

- Dodavanjem cene $\Theta(\lg n)$ za pozive procedure Binary-Search u liniji 10, dobija se sledeća rekurencija:

  $$T_\infty(n) = T_\infty(3n/4) + \Theta(\lg n)$$

- Za osnovni slučaj raspon je $\Theta(1)$
  - Jer se linije 1-8 izvršavaju u konstantnom vremenu
- Rešenje je $T_\infty(n) = \Theta(\lg^2 n)$, što se metodom smene može lako proveriti

Analiza procedure P-Merge (3/4)

- Rad:
  - Pošto svaki od n elemenata mora biti iskopiran iz niza T u niz A, sledi da je $T_1(n) = \Omega(n)$
  - Drugo, pokažimo da je $T_1(n) = O(n)$
  - Bin. pretraga u liniji 10 uzima $\Theta(\lg n)$ vremena
    - što dominira nad drugim radom izvan rekurzivnih poziva
  - Već pokazano: jedan rekurzivni poziv radi na najviše elemenata $3n/4$, pa je:

    $$T_1(n) = T_1(\alpha n) + T_1((1 - \alpha)n) + \Theta(\lg n)$$

    - Parametar α leži u opsegu $1/4 \leq \alpha \leq 3/4$

Analiza procedure P-Merge (4/4)

- Sledi dokaz da je rešenje ove rekurentne jednačine $T_1(n) = O(n)$
  - Metod zamene: pretpostavimo da je $T_1(n) \le c_1 n - c_2 \lg n$ za neke pozitivne konstante c_1 i c_2. Zamenom:
  - $T_1(n) \le (c_1 \alpha n - c_2 \lg(\alpha n)) + (c_1(1 - \alpha)n - c_2 \lg((1 - \alpha)n)) + \Theta(\lg n)$
  - $= \ldots = c_1 n - c_2 \lg n - (c_2(\lg n + \lg(\alpha(1 - \alpha))) - \Theta(\lg n))$
  - $\le c_1 n - c_2 \lg n$
  - c_2 se izabere dovoljno veliko, tako da član $c_2(\lg n + \lg(\alpha(1 - \alpha)))$ dominira nad $\Theta(\lg n)$
  - c_1 se bira dovoljno veliko da se zadovolji rekurenciju
- Znači $T_1(n) = \Theta(n)$, pa je paralelizam $T_1(n)/T_\infty(n) = \Theta(n/\lg^2 n)$.

II verzija paralelnog algoritma sortiranja sa spajanjem podnizova

- Poziv P-Merge-Sort(*A*, *p*, *r*, *B*, *s*) sortira elemente iz *A*[*p*..*r*] i smešta ih u *B*[*s*..*s*+*r*–*p*]

```
P-Merge-Sort(A, p, r, B, s)
1.n = r – p + 1
2.if n == 1
3.    B[s] = A[p]
4.else  neka je T[1..n] novi niz
5.    q = ⌊(p+r)/2⌋
6.    q' = (q – p) + 1
7.    spawn P-Merge-Sort(A, p, q, T, 1)
8.    P-Merge-Sort(A, q+1, r, T, q'+1)
9.    sync
10.P-Merge(T, 1, q', q'+1, n, B, s)
```

- 1: izračuna broj elemenata *n*
- 2-3: osnovni slučaj
- 4: dodeljuje privremeni niz *T*
- 5: računa indeks *q* niza *A*[*p*..*r*] radi njegove podele na dva podniza *A*[*p*..*q*] i *A*[*q*+1..*r*]
- 6: računa broj elemenata *q'* u prvom podnizu *A*[*p*..*q*]
- 7-8: izmrešćeni i običan rekurzivni poziv
- 10: spaja *T*[1..*q'*] i *T*[*q'*+1..*n*] u izlazni podniz *B*[*s*..*s*+*r*–*p*]

Analiza procedure P-Merge-Sort (1/2)

- Rad:
  - Pošto je rad procedure P-Merge $TPM_1(n) = \Theta(n)$, rad procedure P-Merge-Sort je:

    $$T_1(n) = 2\ T_1(n/2) + TPM_1(n)$$
    $$= 2\ T_1(n/2) + \Theta(n)$$

  - Rešenje je $T_1(n) = \Theta(n \lg n)$, prema drugom slučaju master teoreme
- Raspon:
  - Pošto dva rekurzivna poziva rade logički u paraleli, jedan od njih se može ignorisati

Analiza procedure P-Merge-Sort (2/2)

- Pošto raspon procedure P-Merge iznosi $\Theta(\lg^2 n)$, raspon procedure P-Mereg-Sort je:

  $T_\infty(n) = T_\infty(n/2) + TPM_\infty(n)$

  $= T_\infty(n/2) + \Theta(\lg^2 n)$

- Master teorema ne može. Rešenje $T_\infty(n) = \Theta(\lg^3 n)$, se može proveriti metodom smene

- Paralelizam:
  - $T_1(n)/T_\infty(n) = \Theta(n \lg n)/\Theta(\lg^3 n) = \Theta(n / \lg^2 n)$
  - Paralelizam Merge-Sort-P je bio samo $\Theta(\lg n)$
  - $\Theta(n / \lg^2 n)$ je znatno bolje
    - Deo se može žrtvovati ukrupnjavanjem osnovnog slučaja