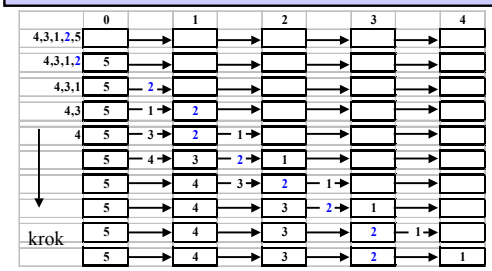
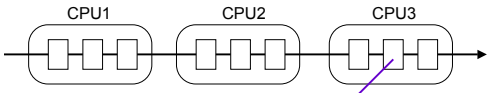
Týden 10
Paralelní řazení

1. vkládáním
2. transpozicí lichý - sudý
3. bitonické
4. s pravidelnými vzorky
5. přihrádkové

Princip paralelního řazení vkládáním (menší propadává doprava)



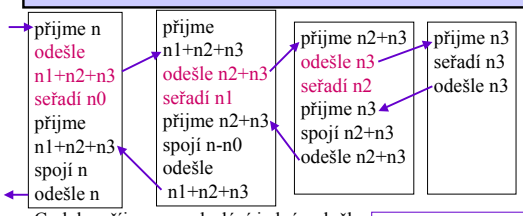
Paralelní řazení vkládáním (Insertion sort)




```
WHILE coming
  SEQ
    input ? next
    IF
      next > highest
        SEQ
          output ! highest
          highest := next
      TRUE
        output ! next
```

Počet procesů
≥
počet klíčů

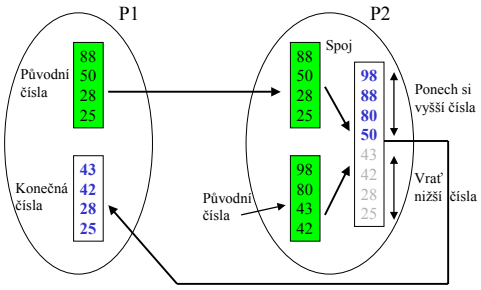
Řazení po blocích dat, n = n0 + n1 + n2 + n3



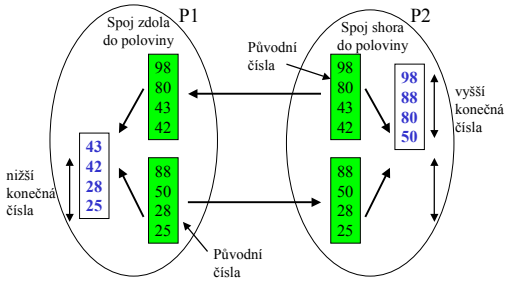
C, doba příjmu a a odeslání jedné položky
S = $q\log q$, doba řazení q položek
M, doba spojení jedné položky
činnosti lze překrýt, ideálně by měly trvat stejně dlouho

Optimální rozdělení
n0 > n1> n2 = n3 ?

Spojování seřazených posloupností (merge)



Spojování seřazených posloupností (2.verze)



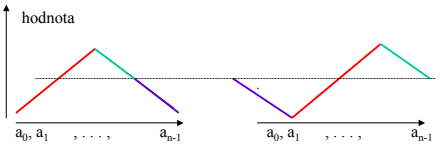
Paralelní řazení lichý-sudý

krok	P0	P1	P2	P3	P4	P5	P6	P7
0	4	2	7	8	5	1	3	6
1	2	4	7	8	1	5	3	6
2	2	4	7	1	8	3	5	6
3	2	4	1	7	3	8	5	6
4	2	1	4	3	7	5	8	6
5	1	2	3	4	5	7	6	8
6	1	2	3	4	5	6	7	8
7	1	2	3	4	5	6	7	8

sudý index		lichý index
recv (P_{i+1})	←	send(P_{i-1})
send (P_{i+1})	→	recv(P_{i-1})
i≥2: recv (P_{i-1})	←	i≤5: send (P_{i+1})
send(P_{i-1})	→	recv(P_{i+1})

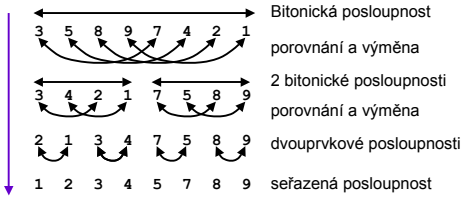
Bitonické posloupnosti

Jsou vytvořeny
- ze 2 částí, jedné narůstající, druhé klesající
- cyklickým posuvem předchozí posloupnosti



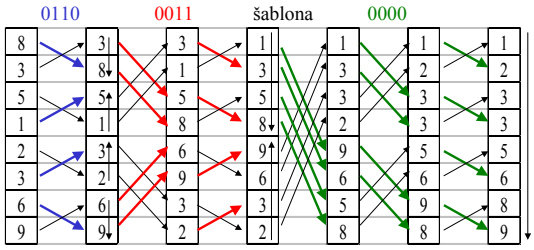
Porovnáním a výměnou prvků a_i a $a_{i+n/2}$ bitonické posloupnosti získáme 2 poloviční bitonické posloupnosti.

Využití bitonických posloupností pro řazení



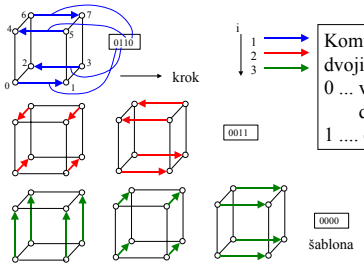
Porovnání a výměna přesunuje menší čísla doleva a větší doprava.
Rekurzivním postupem se prvky seřadí.

Bitonický mergesort P = 8 klíčů int



řádek = CPU; barevné šipky směřují k CPU s vyšším klíčem

Bitonický mergesort na hyperkostce

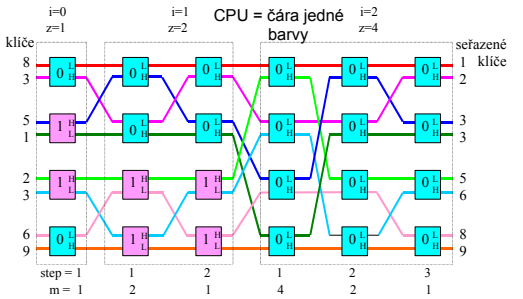


Komunikující dvojice:
0 ... vyšší část do vyššího id,
1 .... opačně.

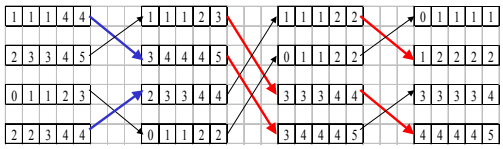
[P/2] INT tmpl: -- binární šablona (template)

```
INT z:
SEQ
  z := 1                -- hodnoty z: 1,2,..,P/2
  SEQ i=0 FOR logP      -- logP šablon pro P procesorů
    INT t,w:
    SEQ                 -- generování šablony tmpl
      SEQ u=0 FOR z
        tmpl[u]:=0      -- počáteční vynulování
      t:=i
      w:=z              -- a nastavení
      WHILE t<=logP-2 -- pro P=8: t=0,1
        SEQ
          SEQ h=0 FOR w
            tmpl[h+w]:=1-tmpl[h] -- komplement tmpl[h]
          t:=t+1
          w:=2*w
      OUTSTRM ! tmpl[P/2-1],...,tmpl[1],tmpl[0]
      z := 2*z
```

HW řadící síť s moduly merge/split



Bitonické spojení P=4 seřazených sekvencí



01 šablona 00

Výkonnost bitonického mergesortu (EREW)

- Každý CPU: sekvenční Quicksort za $\frac{n}{P}\log\frac{n}{P}$ kroků
- Spojení $2\times n/P$ položek (1+2+3+…+ $\log P$)- krát

za $\frac{2n}{P}\frac{\log P(\log P+1)}{2}$ kroků

$$S(n,P)=\frac{n\log n}{\frac{2n}{P}\frac{\log P(1+\log P)}{2}+\frac{n}{P}\log\frac{n}{P}}=P\frac{\log n}{\log n+\log^2 P},$$

$$E=\frac{S}{P}=\frac{\log n}{\log n+\log^2 P}=K \quad \text{pro} \quad \log n=\frac{K}{1-K}\log^2 P$$

$$\rightarrow \quad n=\theta(P^{k\log P})$$

slabá škálovatelnost

Rychlejší bitonický mergesort

- Každý CPU: sekvenční Quicksort za $\frac{n}{P}\log\frac{n}{P}$ kroků
- Spojení jen n/P položek (1+2+3+…+ $\log P$)- krát za

$\frac{n}{P}\frac{(\log P+1)\log P}{2}$ kroků;

$$S(n,P)=\frac{n\log n}{\frac{n}{P}\frac{(1+\log P)\log P}{2}+\frac{n}{P}\log\frac{n}{P}}=P\frac{\log n}{\log n+(\log\sqrt{P})\log(P/2)},$$

$$E=\frac{S}{P}=\frac{\log n}{\log n+(\log\sqrt{P})\log(P/2)}=K$$

škálovatelnost o něco lepší

$$\text{pro} \quad \log n=\frac{K}{1-K}(\log\sqrt{P})\log(P/2)\rightarrow \quad n=\theta((P/2)^{k\log\sqrt{P}})$$

Srovnání řadících algoritmů na EREW PRAM

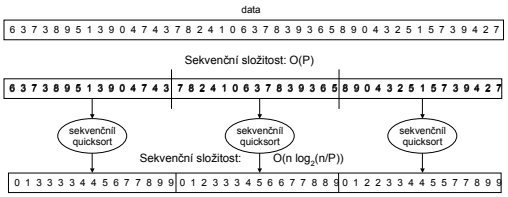
n = 100 000 000, $\log_2$ n = 26,57

	Naivní		Bitonický		Rychlejší bitonický	
P	S	E %	S	E %	S	E %
2	1,9	96	1,9	96	2	100
4	3,5	87	3,5	87	3,8	96
8	5,7	71	6	74	7,1	89
16	8,1	51	10	62	13	81
32	10,2	32	16,5	51	23	72
64	11,6	18	27,2	42	40	63
1024	133	13	215	20	379	37

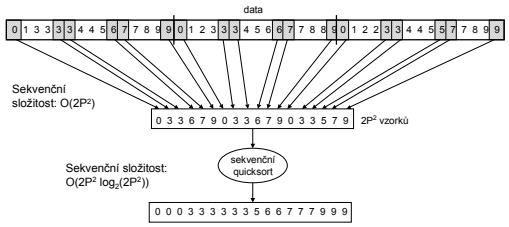
Paralelní řazení pomocí vzorků (PSRS)

- PSRS = Parallel Sorting with Regular Sampling
- Základní myšlenka:
  - Rozděl data do P segmentů stejné velikosti.
  - Seřaď segmenty (např. použitím quicksortu).
  - Obor hodnot dat rozděl na P intervalů (přihrádek).
  - nech si jen jednu přihrádku a data z ostatních přihrádek odešli příslušným procesorům (AAS).
  - Paralelně spoj data ze své a došlých seřazených přihrádek
- Klíčový algoritmus je P-cestné spojování.
- Časová složitost → O(n/P log(n)) for n > P^3.
- Údajně nejrychlejší univerzální paralelní řazení.

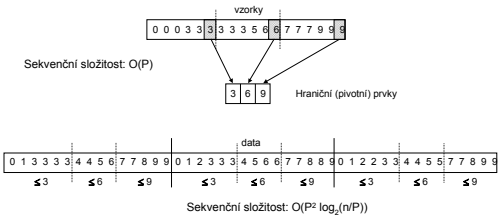
1. Rozděl data do segmentů
2. Paralelně seřaď datové segmenty



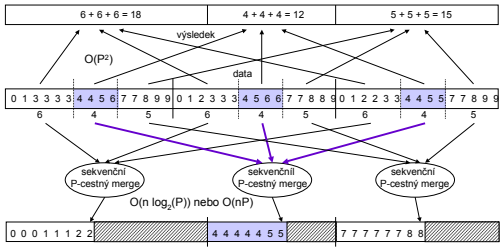
3. Ze seřazených datových segmentů vezmi vzorky ve stejných rozestupech 4. Seřaď vzorky dat



5. Ze seřazených vzorků dat vyber stejně vzdálené hraniční prvky 6. Rozčleň segmenty dat do přihrádek



7. Vypočti velikosti výsledných přihrádek
8. Stejnolehlé přihrádky vždy spojí jeden procesor



Celková sekvenční a paralelní složitost

- Krok 1: O(P) - Rozdělení dat do segmentů.
- Krok 2: $O(\frac{n}{P}\log_2(n/P))$ - Seřazení datových segmentů.
- Krok 3: $O(2P^2)$ - Vzorkování seřazených datových segmentů.
- Krok 4: $O(2P^2 \log_2(2P^2))$ - Seřazení vzorků dat.
- Krok 5: O(P) - Volba pivotů ze seřazených vzorků.
- Krok 6: $O(\frac{P^2}{P}\log_2(n/P))$ - Rozčlenění seřaz.segmentů do přihrádek.
- Krok 7: $O(P^2)$ - Výpočet velikosti přihrádek.
- Krok 8: $O(\frac{n}{P}\log_2(P))$ - Spojení dat do výsledné konfigurace. O(nP) verze by byla jednodušší.

Sekvenční a paralelní časová složitost -pokrač.

- Dva dominantní členy : pokud $2P^2$ **<<** n, téměř celý čas se stráví v krocích 2 a 8:
  - Krok 2: $O(n \log_2(n/P))$ - Řazení datových segmentů.
  - Krok 8: $O(n \log_2(P)$ - Spojení přihrádek dat.
- Při pevném P a n $\rightarrow \infty$, sekv.složitost $\rightarrow O(n \log_2(n))$.
- Oba extrémy (P = 1, P = n) dávají $O(n \log_2(n))$.
- Celkový efekt závisí na násobící konstantě.
- Paralelní složitost

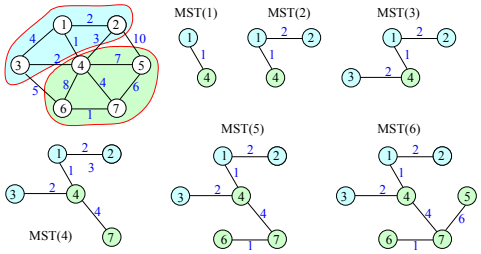
$$O(\tfrac{n}{P}\log\tfrac{n}{P}) + O(\tfrac{n}{P}\log P) \approx O(\tfrac{n}{P}\log n)$$

Paralelní řazení s přihrádkami (bucket sort)

- Opačné pořadí akcí než u PSRS. Data se rozdělí procesorům po intervalech hodnot, teprve pak lokální řazení.
- Základní myšlenka:
  - Rozděl data po segmentech stejné velikosti P procesorům.
  - Obor hodnot dat rozděl na P intervalů.
  - Každý procesor: roztřídí svoje data do P přihrádek (intervalů),
  - nechá si jen jednu přihrádku a data z ostatních přihrádek odešle příslušným procesorům (AAS).
  - seřadí data ve své přihrádce spolu s daty od kolegů.
- Přihrádky budou obsahovat stejný počet klíčů jen při rovnoměrném rozložení hodnot výchozích dat.

Minimální kostra grafu (Min. Spanning Tree)

Je dán neorientovaný graf G, hrany opatřeny vahami.
Kostra grafu G je podgraf ve tvaru stromu obsahující všechny uzly.
Minimální kostra je kostra s nejmenším součtem vah svých hran.
Sekvenční složitost: $O(n^2)$ pro n uzlů.
Uzly grafu jsou rozděleny mezi procesory, MST(0) = root ;

```
for (i=1;i<n;i++) { # tolik kroků, kolik je uzlů (bez src)
   Všechny procesory: najdi uzel nejbližší k MST(i-1) a zapiš délku cesty
   do sdílené paměti;
   Všichni redukují operací minimum předchozí nejkratší cesty na jedno
   globální minimum za log P kroků; #všichni je budou znát najednou
   cpu[0]: aktualizuj MST(i); # zapíše do MST ve sdílené paměti právě
                              # nalezený uzel nejbližší k MST(i-1)
   barrier(id);
}
```

Minimální kostra grafu (MST) - pokrač.