
Paralelné programovanie

Návrh paralelných algoritmov

Bc. št. prog. Informatika - 2010/2011

Ing. Michal Čerňanský, PhD.

Fakulta informatiky a
informačných technológií,
STU Bratislava

Prehľad tém

- Úvod do paralelných algoritmov
 - Úlohy a dekompozícia
 - Procesy a mapovanie
 - Procesy vs. procesory
 - Techniky dekompozície
 - Rekurzívna dekompozícia
 - Dátová dekompozícia
 - Exploratívna dekompozícia
 - Hybridná dekompozícia
 - Vlastnosti úloh a interakcie medzi úlohami
 - Generovanie úloh, zrnitosť a kontext
 - Vlastnosti interakcie medzi úlohami
-

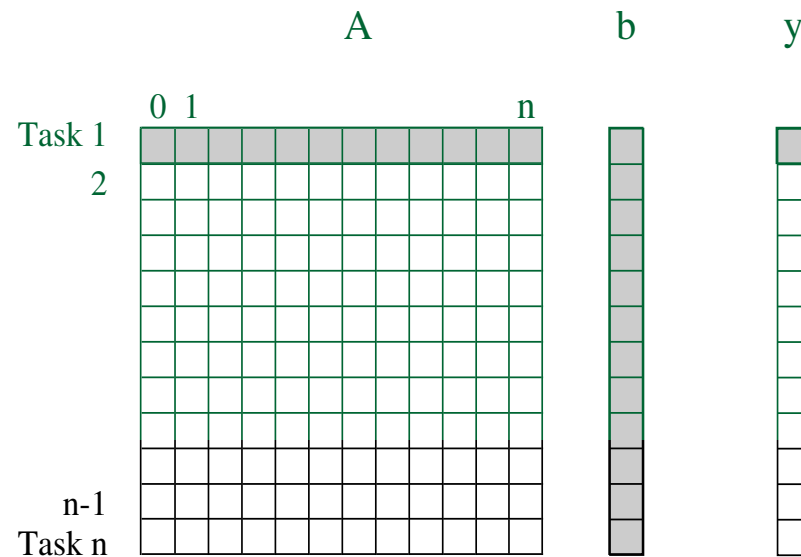
Prehľad tém

- Mapovanie a techniky vyrovnávania záťaže
 - Statické a dynamické mapovanie
 - Metódy na minimalizáciu réžie spojenej s interakciou
 - Maximalizácia dátovej lokálnosti
 - Minimalizácia obsadzovania a úzkych hrdiel
 - Prekrývanie komunikácie a výpočtov
 - Skupinová vs. bod-bod komunikácia
 - Návrhové modely pre paralelné architektúry
 - Dátovo-paralelné, bazén úloh, graf úloh, nadriadený-podriadený, prúdová linka a hybridné modely
-

Úvod: dekompozícia, úlohy a grafy závislosti

- Prvý krok - dekomponovať problém na úlohy vykonateľné súbežne
 - Viaceré možnosti dekompozície
 - Úlohy rovnakej, rôznej a tiež nekončiacej veľkosti
 - Dekompozícia môže byť znázornená formou orientovaného grafu s uzlami reprezentujúcimi úlohy a hranami znázorňujúcimi závislosť jednej úlohy od inej – graf závislosti
-

Príklad: násobenie matice s vektorom



- Výpočet každého prvku výstupného vektora y je nezávislé na iných prvkoch – možnosť dekomponovať násobenie na n nezávislých úloh.
- Úlohy zdieľajú pole b , ale žiadne závislosti medzi úlohami, veľkosť úloh je rovnaká.

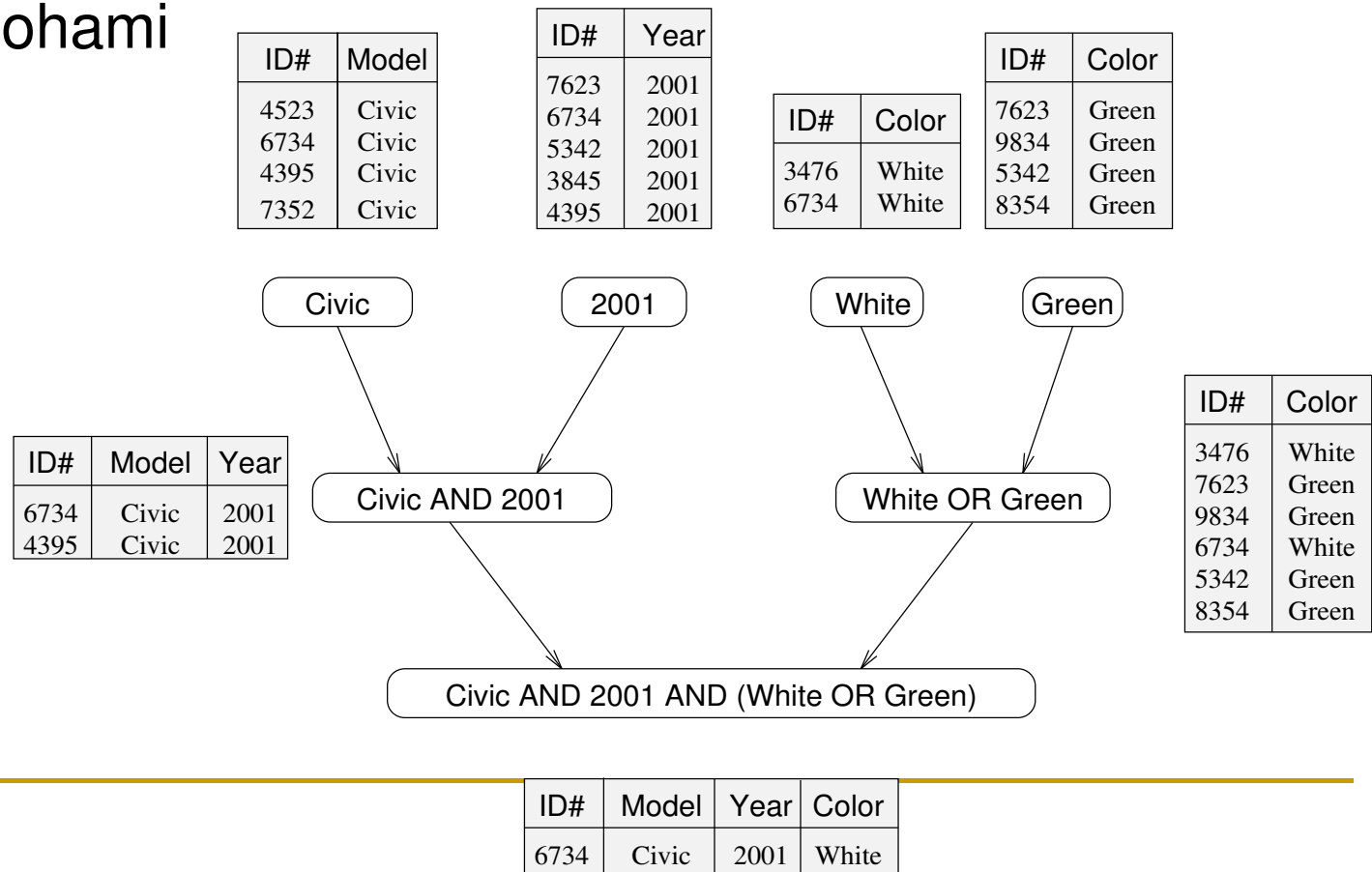
Príklad: Spracovanie DB požiadavky

Požiadavka: MODEL = ``CIVIC" AND YEAR = 2001 AND
(COLOR = ``GREEN" OR COLOR = ``WHITE)

ID#	Model	Year	Color	Dealer	Price
4523	Civic	2002	Blue	MN	\$18,000
3476	Corolla	1999	White	IL	\$15,000
7623	Camry	2001	Green	NY	\$21,000
9834	Prius	2001	Green	CA	\$18,000
6734	Civic	2001	White	OR	\$17,000
5342	Altima	2001	Green	FL	\$19,000
3845	Maxima	2001	Blue	NY	\$22,000
8354	Accord	2000	Green	VT	\$18,000
4395	Civic	2001	Red	CA	\$17,000
7352	Civic	2002	Red	WA	\$18,000

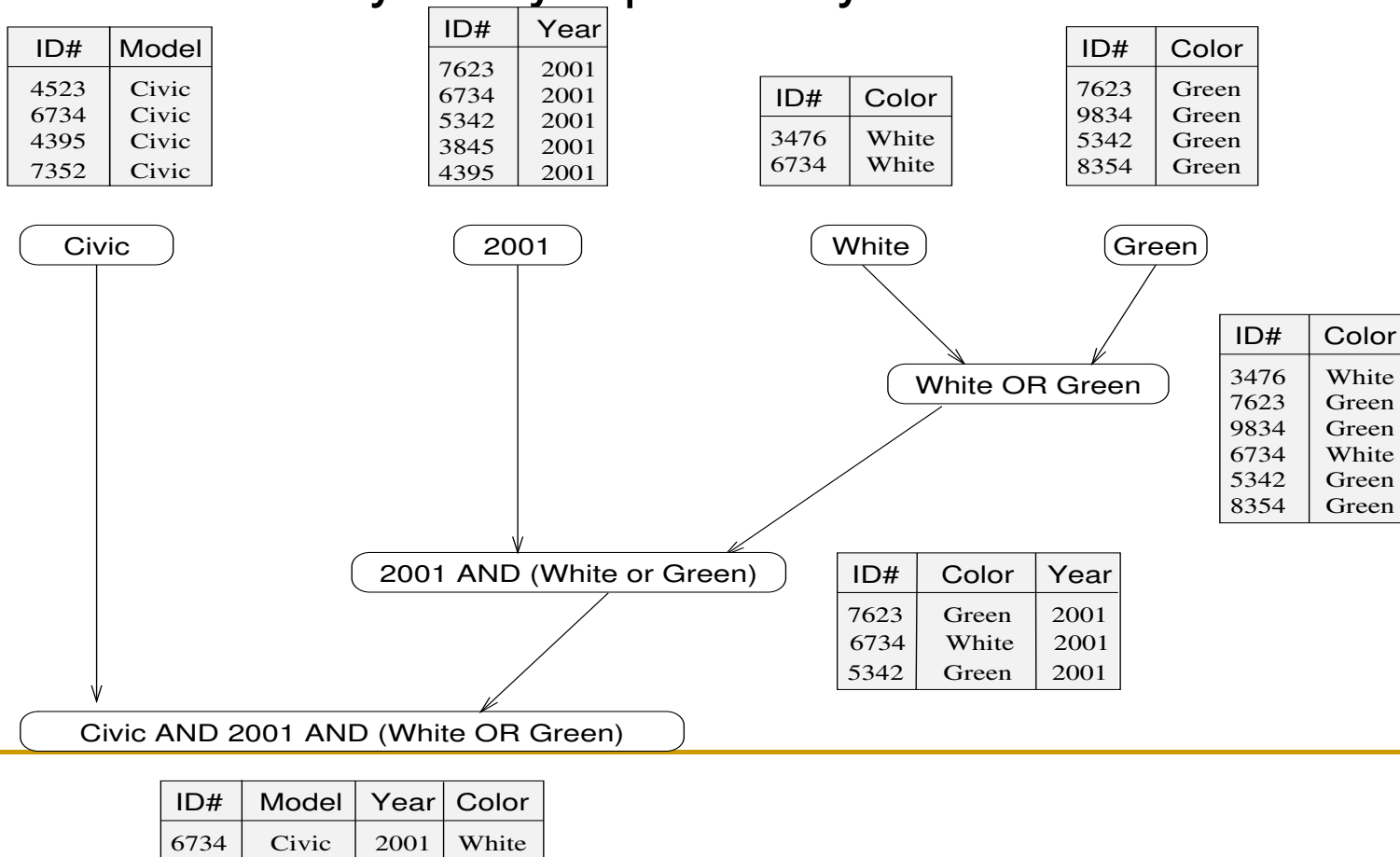
Príklad: Spracovanie DB požiadavky

- Spracovanie požiadavky môže byť rozdelené do viacerých úloh. Hrany v grafe znázorňujú závislosť medzi úlohami



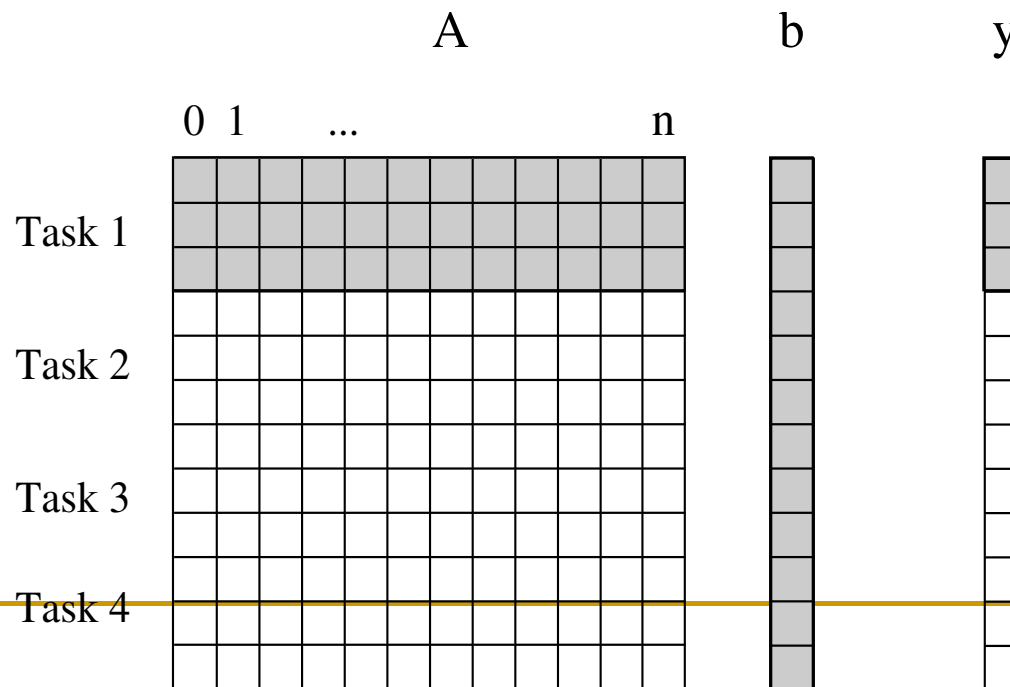
Príklad: Spracovanie DB požiadavky

- Iná dekompozícia úlohy. Rôzne dekompozície môžu viesť k rôzne výkonným paralelným riešeniam



Zrnitosť dekompozície na úlohy

- Počet úloh, na ktoré je problém dekomponovaný určuje zrornosť (Granularity) riešenia
- Veľa úloh – jemná zrornosť (Fine-grained)
- Hrubozrnné (Coarse-grained) rozdelenie násobenia



Stupeň súbežnosti (Concurrency)

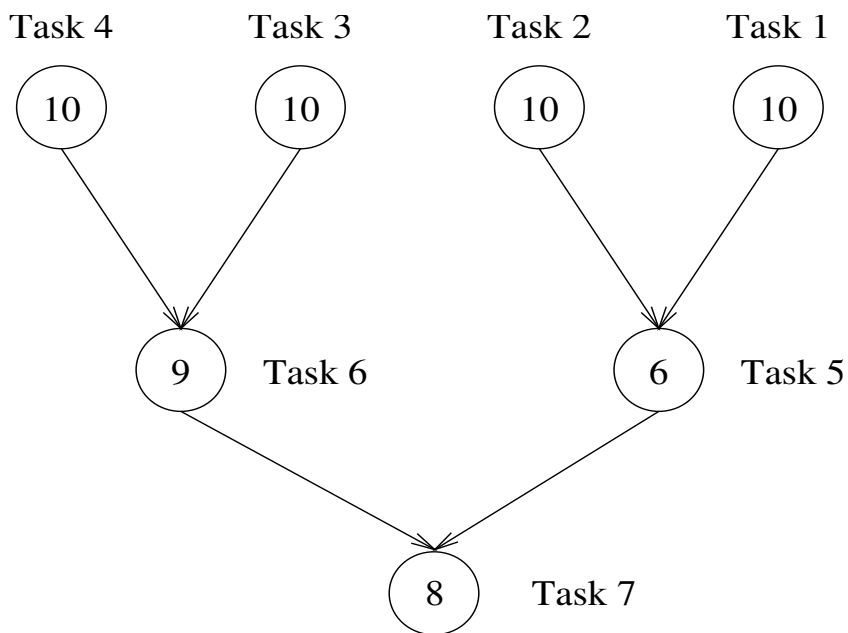
- Počet naraz vykonateľných úloh – stupeň súbežnosti
 - Počet naraz vykonateľných úloh sa v priebehu programu mení
 - Maximálny stupeň súbežnosti – max. počet úloh, kt. sú naraz vykonateľné v ľubovoľnom čase počas behu programu
 - Priemerný stupeň súbežnosti – zohľadnenie času, priemerný počet naraz vykonateľných úloh
 - Stupeň súbežnosti rastie so zmenšujúcou sa zrnitosťou
-

Dĺžka kritickej cesty

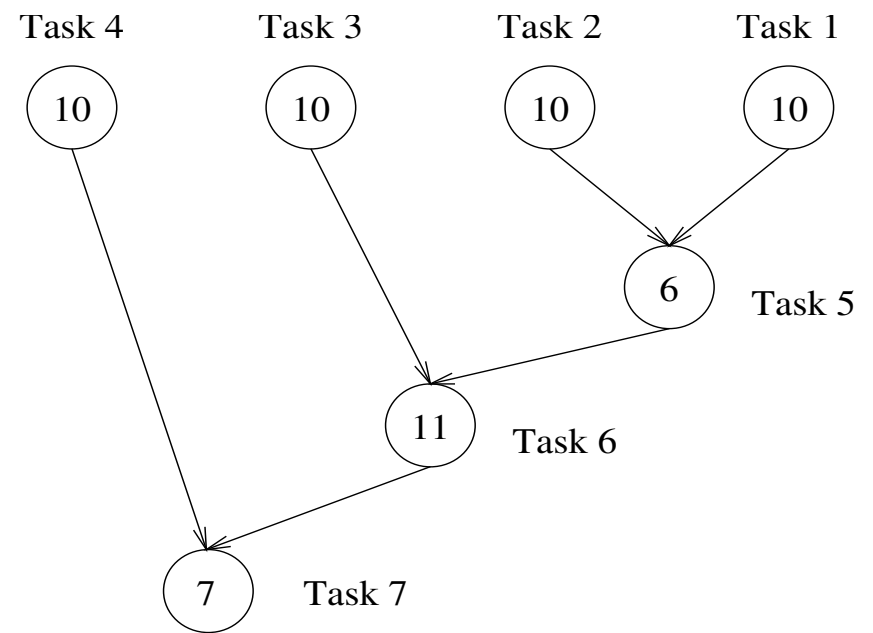
- Cesta po orientovaných hranách v grafe závislosti reprezentuje postupnosť úloh, ktoré musia byť vykonané jedna po druhej v danom poradí
 - Najdlhšia taká cesta určuje najkratší možný čas vykonania paralelného programu
 - Dĺžka najdlhšej cesty v grafe závislosti sa volá dĺžka kritickej cesty
-

Dĺžka kritickej cesty

- Grafy závislosti pre dve dekompozície DB dopytu.
 - Dĺžka kritickej cesty
 - Najkratší čas paralelného vykonania
 - Maximálny stupeň súbežnosti



(a)



(b)

Limitácie paralelného spracovania

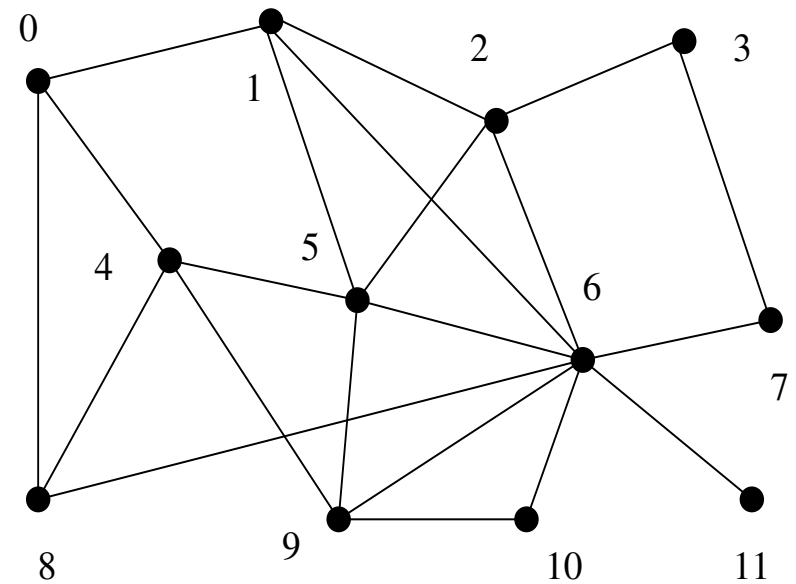
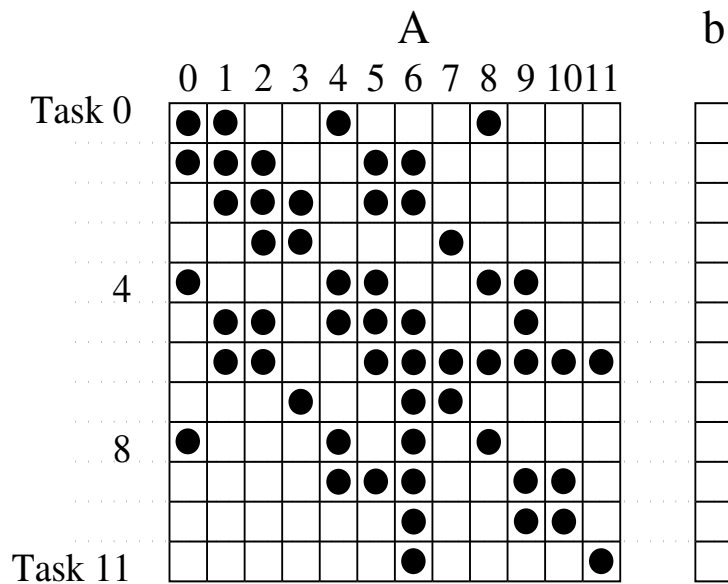
- Čas paralelného spracovania nemôže byť ľubovoľne zmenšovaný zjemňovaním zrnitosti dekompozície
 - Ohraničenie granularity, napr. násobenie matice a vektora – max. n^2 súbežných úloh
 - Súbežné úlohy tiež môžu vyžadovať vzájomnú výmenu údajov, výsledkom je komunikačná réžia
 - Rovnováha medzi stupňom zrnitosti dekompozície a k tomu prislúchajúcou réziou často určuje výkonnostné ohraničenie
-

Graf interakcie medzi úlohami

- Dekomponované úlohy si zvyčajne potrebujú vymieňať údaje
 - Triviálne násobenie matice s vektorom – ak vektor **b** nie je replikovaný, je potrebné komunikovať prvky vektora
 - Graf úloh (uzly) a ich interakcií-výmeny údajov (hrany) sa nazýva graf interakcie medzi úlohami (Task Interaction Graph)
 - Graf interakcie – dátová závislosť
 - Graf závislosti – riadiaca závislosť
 -
-

Príklad: graf interakcie medzi úlohami

- Násobenie riedkej matice **A** s vektorom **b**
 - Násobenie každého riadku samostatná úloha
 - Iba nenulové prvky matice **A** sa zúčastňujú výpočtu
 - Ak aj vektor **b** je rozdelený medzi úlohy



Graf interakcie, zrnitosť a komunikácia

- Jemnejšia zrnitosť – vyššia réžia
 - Násobenie riedkej matice s vektorom
 - Jedna jednotka času „stratená“ spracovaním interakcie
 - Jedna jednotka času „stratená“ výpočtom
 - Uzol 0 – samostatná úloha, 1 jednotka výpočet a 3 jednotky komunikácia
 - Uzly 0,4 a 5 jedna úloha – 3 jednotky výpočet a 4 jednotky komunikácia – lepší pomer výpočtu ku komunikácií
-

Procesy a mapovanie úloh na procesy

- Vo všeobecnosti počet úloh je vyšší ako počet výpočtových spracovateľských elementov – procesov
 - Paralelný algoritmus musí riešiť mapovanie úloh na procesy
 - Procesy vs. procesory
 - Zvyčajne nie je možné priame mapovanie úloh na procesory
 - Úlohy na procesy a OS mapuje procesy na procesory
 - Proces – voľné pomenovanie súboru úloh a k nim prislúchajúcich údajov
-

Procesy a mapovanie úloh na procesy

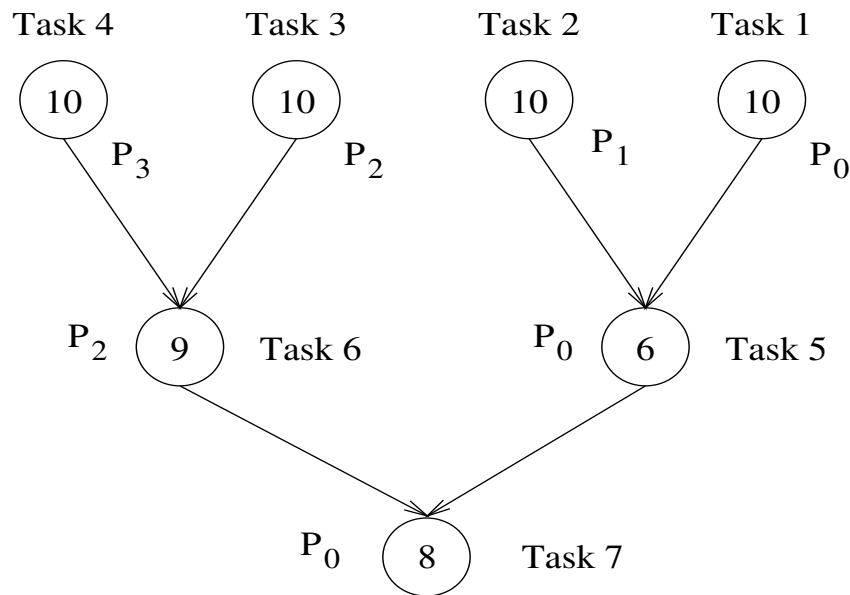
- Vhodné mapovanie úloh na procesy – kritické pre výkon paralelného algoritmu
 - Mapovanie je určené grafom závislosti aj grafom interakcie
 - Grafy závislosti môžu pomôcť pri rovnomernom rozložení práce medzi procesy v každom čase (minimálne čakanie a optimálne zaťaženie)
 - Grafy interakcie môžu pomôcť pri minimalizácii interakcií medzi procesmi (minimálna komunikácia)
-

Procesy a mapovanie úloh na procesy

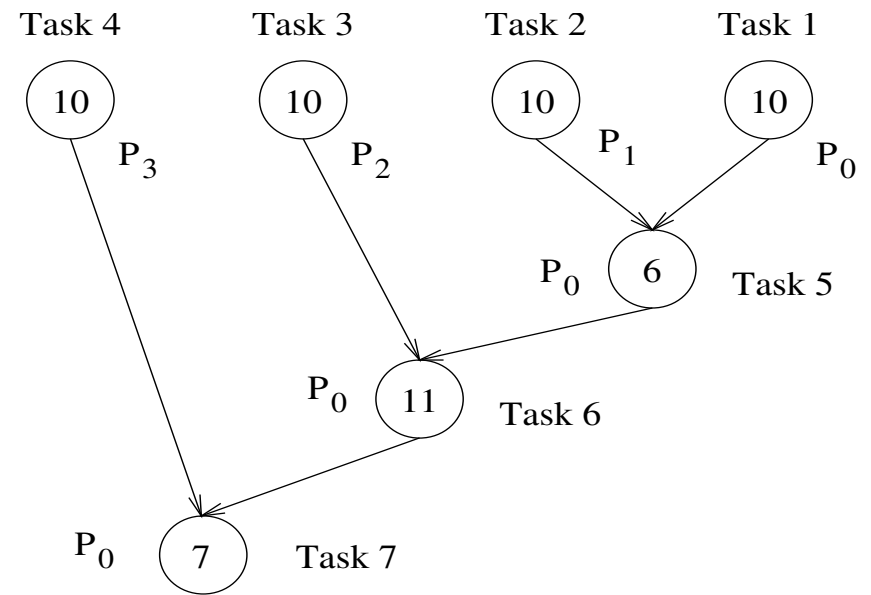
- Vhodné mapovanie musí minimalizovať čas paralelného vykonania programu:
 - Mapovanie nezávislých úloh na rôzne procesy
 - Priradenie úloh na kritickej ceste procesom hneď ako je to možné
 - Minimalizácia interakcie medzi procesmi mapovaním úloh s hustou komunikáciou na rovnaký proces
 - Často protichodné ciele, napr. všetky úlohy na jediný proces zredukuje interakciu, ale žiadne zrýchlenie sa nedosiahne
-

Procesy a mapovanie - príklad

- Mapovanie úloh spracovania DB dopytu na procesy (žiadne dve úlohy na jednej úrovni nie sú spracované v rovnakom procese)



(a)



(b)

Techniky dekompozície

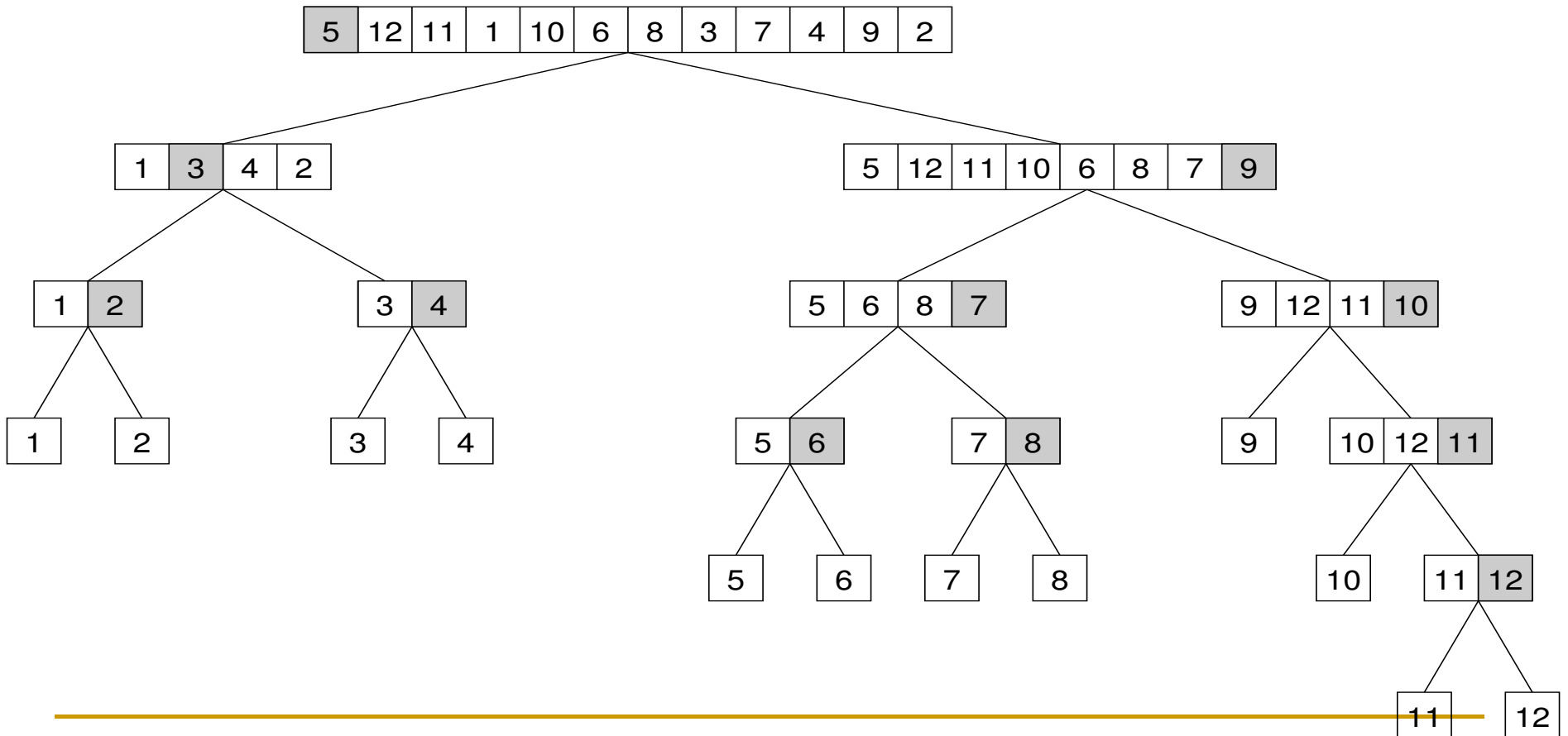
- Ako teda dekomponovať úlohu na podúlohy?
 - Bežné techniky použiteľné na širokú triedu problémov
 - Rekurzívna dekompozícia
 - Dátová dekompozícia
 - Exploratívna dekompozícia
 - Špekulatívna dekompozícia
-

Rekurzívna dekompozícia - príklad

- Vo všeobecnosti vhodné pre problémy riešené stratégiou „rozdeľuj a panuj“.
 - Problém je najskôr dekomponovaný na množinu podproblémov.
 - Podproblémy sú potom rekurzívne dekomponované až kým nie je dosiahnutá požadovaná zrnitosť.
-

Rekurzívna dekompozícia - príklad

- Klasický „rozdeľuj a panuj“ problém - quicksort



Rekurzívna dekompozícia - príklad

- Hľadanie minima v zozname prvkov
- Riešenie stratégiou „rozdeľuj a panuj“
- Jednoduché sekvenčné riešenie:

```
procedure SERIAL_MIN (A, n)
```

```
begin
```

```
  min = A[0];
```

```
  for i := 1 to n - 1 do
```

```
    if (A[i] < min) min := A[i];
```

```
  endfor;
```

```
  return min;
```

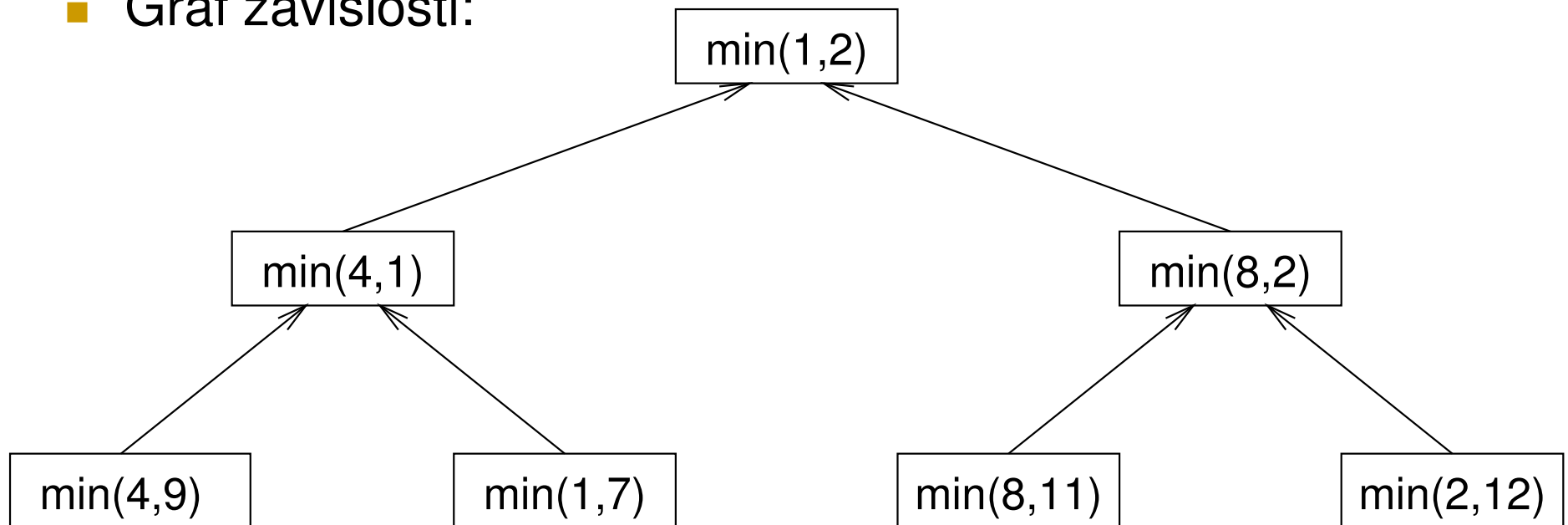
```
end SERIAL_MIN
```

Rekurzívna dekompozícia - príklad

```
procedure RECURSIVE_MIN (A, n)  
begin  
  if ( n = 1 ) then min := A [0] ;  
  else  
    lmin := RECURSIVE_MIN ( A, n/2 );  
    rmin := RECURSIVE_MIN ( &( A[n/2]), n - n/2 );  
    if (lmin < rmin) then min := lmin;  
    else min := rmin;  
    endif;  
  endif;  
  return min;  
end RECURSIVE_MIN
```

Rekurzívna dekompozícia - príklad

- Prirodzené dekomponovanie problému rekurzívnou dekompozíciou
- Pole čísel {4, 9, 1, 7, 8, 11, 2, 12}
- Graf závislosti:



Dátová dekompozícia

- Identifikácia údajov, nad ktorými sa výpočet realizuje.
 - Rozdelenie údajov medzi rôzne úlohy.
 - Rozdelenie zahŕňa aj dekompozíciu problému
 - Možnosť rozdeliť údaje rôznymi spôsobmi – kritické vzhľadom na výkonnosť paralelného algoritmu
-

Dátová dekompozícia – podľa výstupných údajov

- Často každý prvok výstupných údajov môže byť vypočítaný nezávisle na ostatných (jednoducho ako funkcia vstupov)
 - Rozdelenie výstupu medzi úlohy prirodzene dekomponuje problém.
-

Dekompozícia podľa výstupných údajov - príklad

- Násobenie dvoch $n \times n$ matíc A a B do výslednej matice C – rozdelenie do štyroch úloh:

$$\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \cdot \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix} \rightarrow \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}$$

Task 1: $C_{1,1} = A_{1,1}B_{1,1} + A_{1,2}B_{2,1}$

Task 2: $C_{1,2} = A_{1,1}B_{1,2} + A_{1,2}B_{2,2}$

Task 3: $C_{2,1} = A_{2,1}B_{1,1} + A_{2,2}B_{2,1}$

Task 4: $C_{2,2} = A_{2,1}B_{1,2} + A_{2,2}B_{2,2}$

Dekompozícia podľa výstupných údajov - príklad

- Dekompozícia nie je jednoznačná, ďalšie dve možnosti:

Dekompozícia I	Dekompozícia II
Task 1: $C_{1,1} = A_{1,1} B_{1,1}$	Task 1: $C_{1,1} = A_{1,1} B_{1,1}$
Task 2: $C_{1,1} = C_{1,1} + A_{1,2} B_{2,1}$	Task 2: $C_{1,1} = C_{1,1} + A_{1,2} B_{2,1}$
Task 3: $C_{1,2} = A_{1,1} B_{1,2}$	Task 3: $C_{1,2} = A_{1,2} B_{2,2}$
Task 4: $C_{1,2} = C_{1,2} + A_{1,2} B_{2,2}$	Task 4: $C_{1,2} = C_{1,2} + A_{1,1} B_{1,2}$
Task 5: $C_{2,1} = A_{2,1} B_{1,1}$	Task 5: $C_{2,1} = A_{2,2} B_{2,1}$
Task 6: $C_{2,1} = C_{2,1} + A_{2,2} B_{2,1}$	Task 6: $C_{2,1} = C_{2,1} + A_{2,1} B_{1,1}$
Task 7: $C_{2,2} = A_{2,1} B_{1,2}$	Task 7: $C_{2,2} = A_{2,1} B_{1,2}$
Task 8: $C_{2,2} = C_{2,2} + A_{2,2} B_{2,2}$	Task 8: $C_{2,2} = C_{2,2} + A_{2,2} B_{2,2}$

Dekompozícia podľa výstupných údajov - príklad

- Problém určenia početností

(a) Transactions (input), itemsets (input), and frequencies (output)

Database Transactions	A, B, C, E, G, H	Itemsets	A, B, C	Itemset Frequency	1
	B, D, E, F, K, L		D, E		3
	A, B, F, H, L		C, F, G		0
	D, E, F, H		A, E		2
	F, G, H, K,		C, D		1
	A, E, F, K, L		D, K		2
	B, C, D, G, H, L		B, C, F		0
	G, H, L		C, D, K		0
	D, E, F, K, L				
	F, G, H, L				

(b) Partitioning the frequencies (and itemsets) among the tasks

Database Transactions	A, B, C, E, G, H	Itemsets	A, B, C	Itemset Frequency	1
	B, D, E, F, K, L		D, E		3
	A, B, F, H, L		C, F, G		0
	D, E, F, H		A, E		2
	F, G, H, K,				
	A, E, F, K, L				
	B, C, D, G, H, L				
	G, H, L				
	D, E, F, K, L				
	F, G, H, L				

task 1

Database Transactions	A, B, C, E, G, H	Itemsets	C, D	Itemset Frequency	1
	B, D, E, F, K, L		D, K		2
	A, B, F, H, L		B, C, F		0
	D, E, F, H		C, D, K		0
	F, G, H, K,				
	A, E, F, K, L				
	B, C, D, G, H, L				
	G, H, L				
	D, E, F, K, L				
	F, G, H, L				

task 2

Dekompozícia podľa výstupných údajov - príklad

- Ak je databáza replikovaná medzi procesmi, každá úloha môže byť vykonaná bez komunikácie.
 - Ak je databáza rozdelená medzi procesy, každá úloha vykoná výpočet čiastočných početností. Tieto početnosti sa nakoniec agregujú do výsledných početností.
-

Dekompozícia podľa vstupných údajov

- Vo všeobecnosti použiteľné, ak každý výstup môže byť prirodzene vypočítaný ako funkcia vstupov
 - V mnohých prípadoch jediný prirodzený spôsob dekompozície, keďže výstup nie je dopredu známi (napr. problém hľadania minimálneho prvku poľa).
 - Úloha je asociovaná s každou časťou vstupných dát. Väčšinu výpočtu úloha realizuje nad svojou časťou dát.
 - Nasledujúce spracovanie skombinuje čiastkové výsledky.
-

Dekompozícia podľa vstupných údajov - príklad

- V určovaní početností v DB – rozdelenie transakcii

Partitioning the transactions among the tasks

Database Transactions	A, B, C, E, G, H	Itemsets	A, B, C	Itemset Frequency	1
	B, D, E, F, K, L		D, E		2
	A, B, F, H, L		C, F, G		0
	D, E, F, H		A, E		1
	F, G, H, K,		C, D		0
			D, K		1
			B, C, F		0
			C, D, K		0

task 1

Database Transactions		Itemsets	A, B, C	Itemset Frequency	0
			D, E		1
			C, F, G		0
	A, E, F, K, L		A, E		1
	B, C, D, G, H, L		C, D		1
	G, H, L		D, K		1
	D, E, F, K, L		B, C, F		0
	F, G, H, L		C, D, K		0

task 2

Dekompozícia podľa vstupných a výstupných údajov - príklad

Partitioning both transactions and frequencies among the tasks

Database Transactions	A, B, C, E, G, H	Itemsets	A, B, C	Itemset Frequency	1
	B, D, E, F, K, L		D, E		2
	A, B, F, H, L		C, F, G		0
	D, E, F, H		A, E		1
	F, G, H, K,				

task 1

Database Transactions	A, B, C, E, G, H	Itemsets		Itemset Frequency	
	B, D, E, F, K, L				
	A, B, F, H, L				
	D, E, F, H		C, D		0
	F, G, H, K,		D, K		1
	B, C, F	0			
	C, D, K	0			

task 2

Database Transactions		Itemsets	A, B, C	Itemset Frequency	0
			D, E		1
			C, F, G		0
	A, E, F, K, L		A, E		1
	B, C, D, G, H, L				
G, H, L					
D, E, F, K, L					
F, G, H, L					

task 3

Database Transactions	A, E, F, K, L	Itemsets		Itemset Frequency	
	B, C, D, G, H, L		C, D		1
	G, H, L		D, K		1
	D, E, F, K, L		B, C, F		0
	F, G, H, L		C, D, K		0

task 4

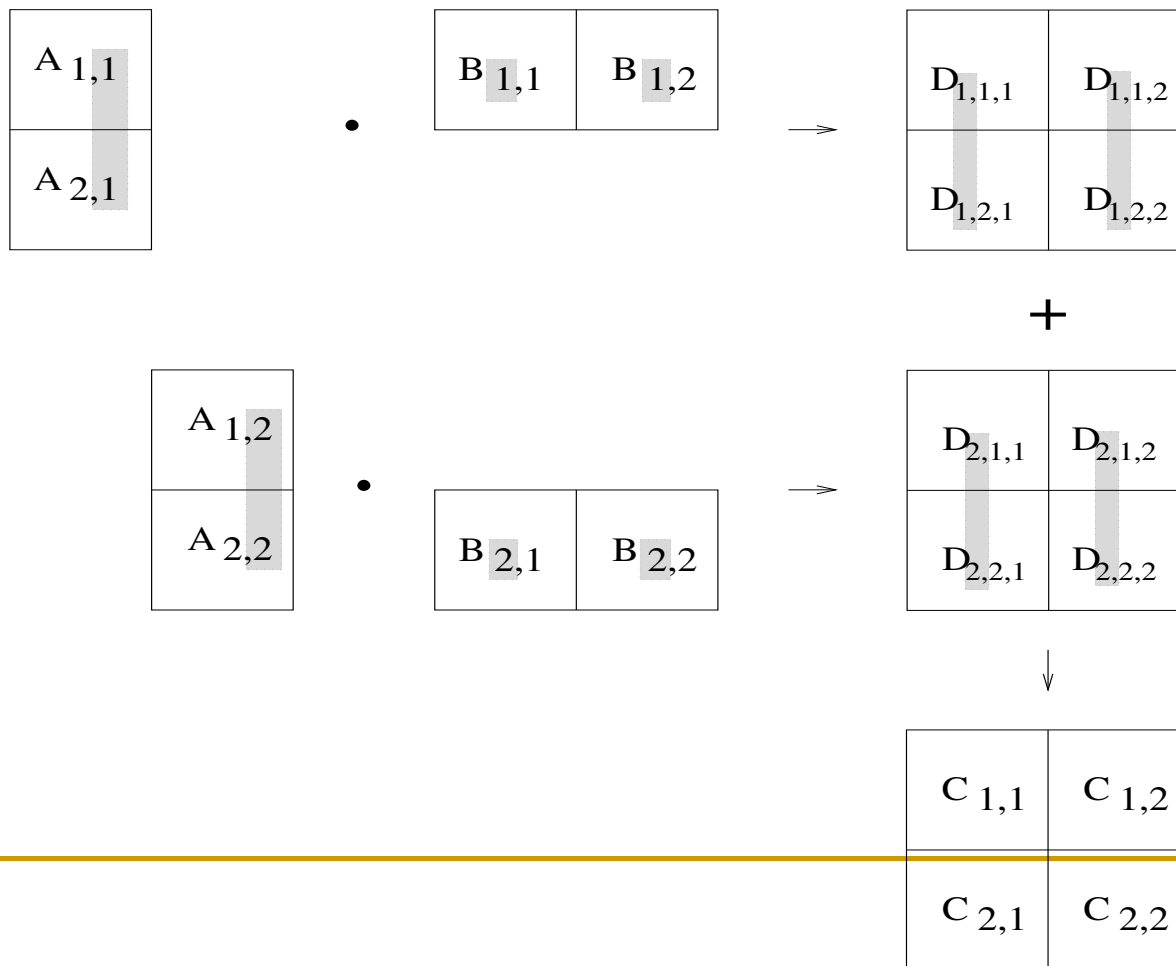
Dekompozícia podľa medzivýsledkov

- Výpočet môže byť často považovaný za transformáciu vstupov na výstupy
 - V takomto prípade je často užitočné využiť medzivýsledky ako základ pre dekompozíciu
-

Dekompozícia podľa medzivýsledkov

- príklad

- Násobenie matic – medzivýsledky v maticiach **D**.



Dekompozícia podľa medzivýsledkov

- príklad

Fáza I – 8 úloh

$$\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \cdot \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix} \rightarrow \begin{pmatrix} \begin{pmatrix} D_{1,1,1} & D_{1,1,2} \\ D_{1,2,2} & D_{1,2,2} \end{pmatrix} \\ \begin{pmatrix} D_{2,1,1} & D_{2,1,2} \\ D_{2,2,2} & D_{2,2,2} \end{pmatrix} \end{pmatrix}$$

Fáza II – 4 úlohy

$$\begin{pmatrix} D_{1,1,1} & D_{1,1,2} \\ D_{1,2,2} & D_{1,2,2} \end{pmatrix} + \begin{pmatrix} D_{2,1,1} & D_{2,1,2} \\ D_{2,2,2} & D_{2,2,2} \end{pmatrix} \rightarrow \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}$$

Task 01: $D_{1,1,1} = A_{1,1} B_{1,1}$

Task 02: $D_{2,1,1} = A_{1,2} B_{2,1}$

Task 03: $D_{1,1,2} = A_{1,1} B_{1,2}$

Task 04: $D_{2,1,2} = A_{1,2} B_{2,2}$

Task 05: $D_{1,2,1} = A_{2,1} B_{1,1}$

Task 06: $D_{2,2,1} = A_{2,2} B_{2,1}$

Task 07: $D_{1,2,2} = A_{2,1} B_{1,2}$

Task 08: $D_{2,2,2} = A_{2,2} B_{2,2}$

Task 09: $C_{1,1} = D_{1,1,1} + D_{2,1,1}$

Task 10: $C_{1,2} = D_{1,1,2} + D_{2,1,2}$

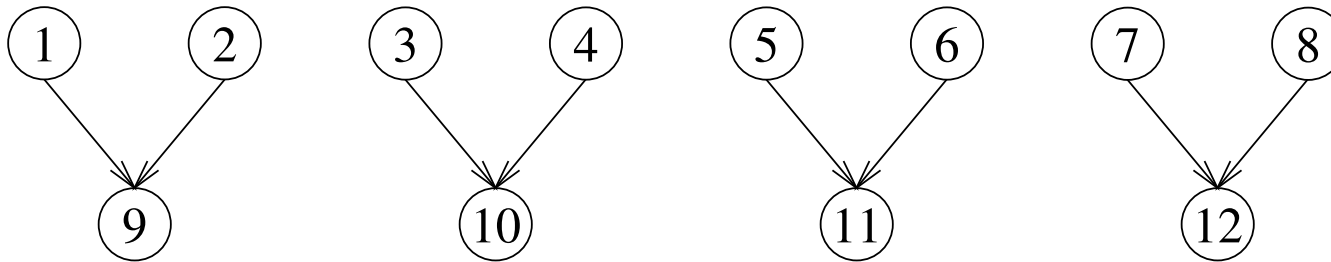
Task 11: $C_{2,1} = D_{1,2,1} + D_{2,2,1}$

Task 12: $C_{2,2} = D_{1,2,2} + D_{2,2,2}$

Dekompozícia podľa medzivýsledkov

- príklad

- Graf závislosti:



Pravidlo „vlastník počítača“

- Pravidlo vlastník počítača (Owner Computes) – proces priradený k nejakému údaju je zodpovedný za všetky výpočty spojené s údajom.
 - Dekompozícia na základe vstupných údajov – všetky výpočty nad vstupnými údajmi sú vykonávané procesom, ktorému prislúchajú vstupné údaje.
 - Dekompozícia na základe výstupných údajov – výstup je vypočítaný procesom, ktorému prislúchajú výstupné údaje.
-

Exploračná dekompozícia

- V mnohých prípadoch dekompozícia problému je spojená s vykonávaním programu.
 - Takéto problémy zvyčajne vyžadujú prehľadávanie stavového priestoru riešení
 - Problémy z tejto oblasti spadajú pod optimalizačné problémy, dokazovanie tvrdení, hranie hier, ...
-

Exploračná dekompozícia - príklad

- Jednoduchá hra (15 puzzle)

1	2	3	4
5	6	7	8
9	10	7	11
13	14	15	12

(a)

1	2	3	4
5	6	7	8
9	10	11	11
13	14	15	12

(b)

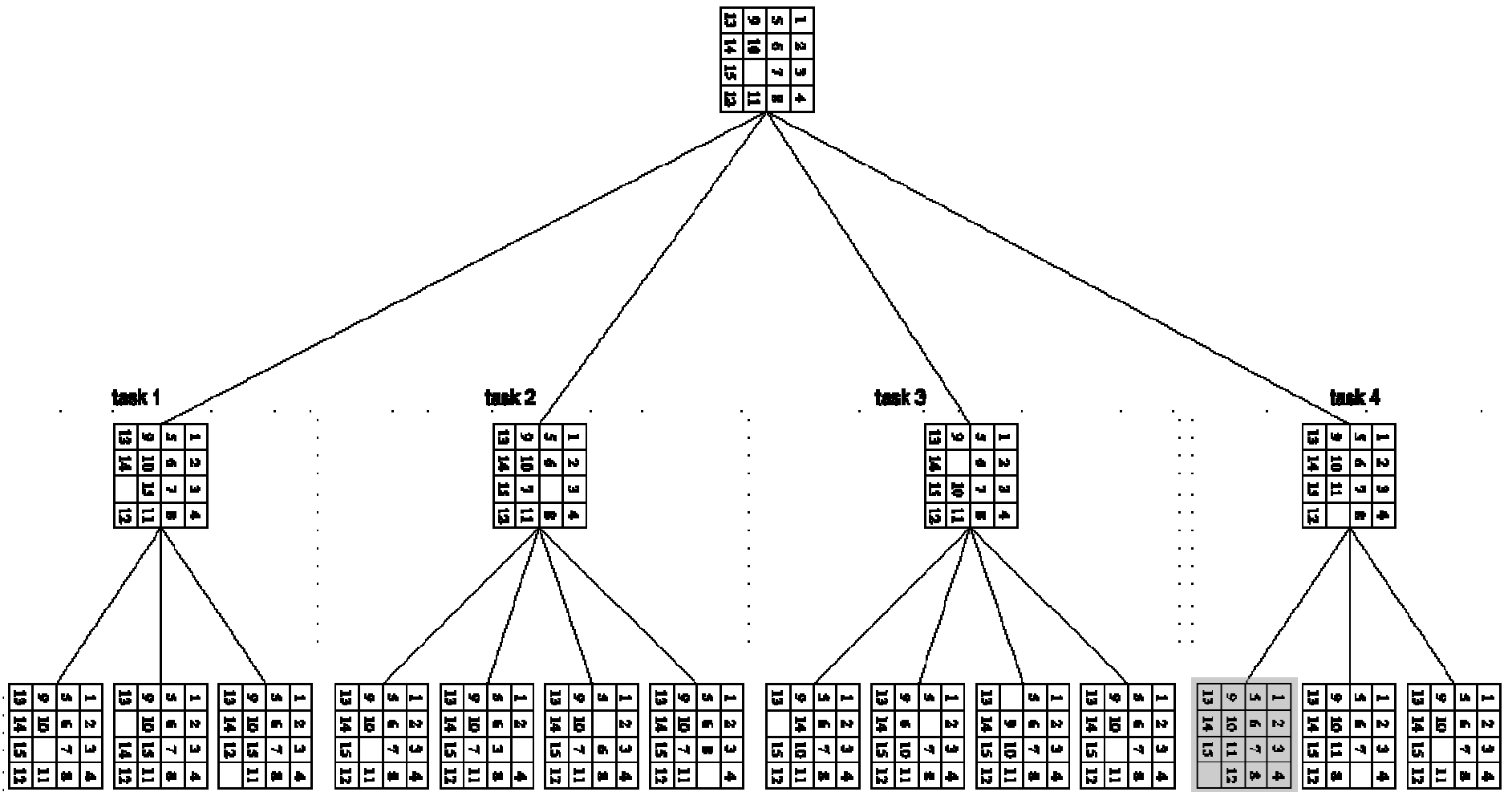
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	12

(c)

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

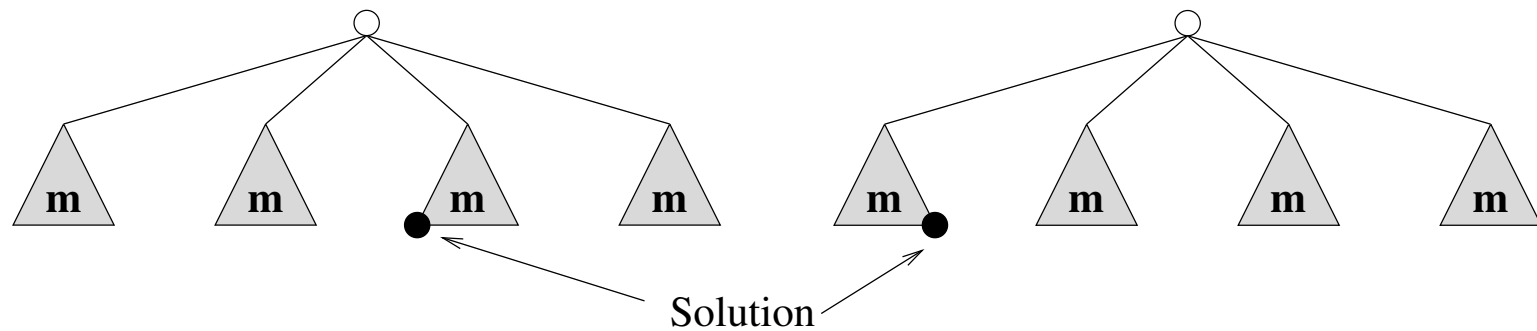
(d)

Exploračná dekompozícia - príklad



Exploračná dekompozícia – anomálne výpočty

- V mnohých prípadoch použitia exploračnej dekompozície – spôsob dekomponovania určuje množstvo práce v paralelnom riešení
- Výsledok môže byť super- lineárny alebo sub-lineárny



Total serial work: $2m+1$
Total parallel work: 1

(a)

Total serial work: m
Total parallel work: $4m$

(b)

Špekulatívna dekompozícia

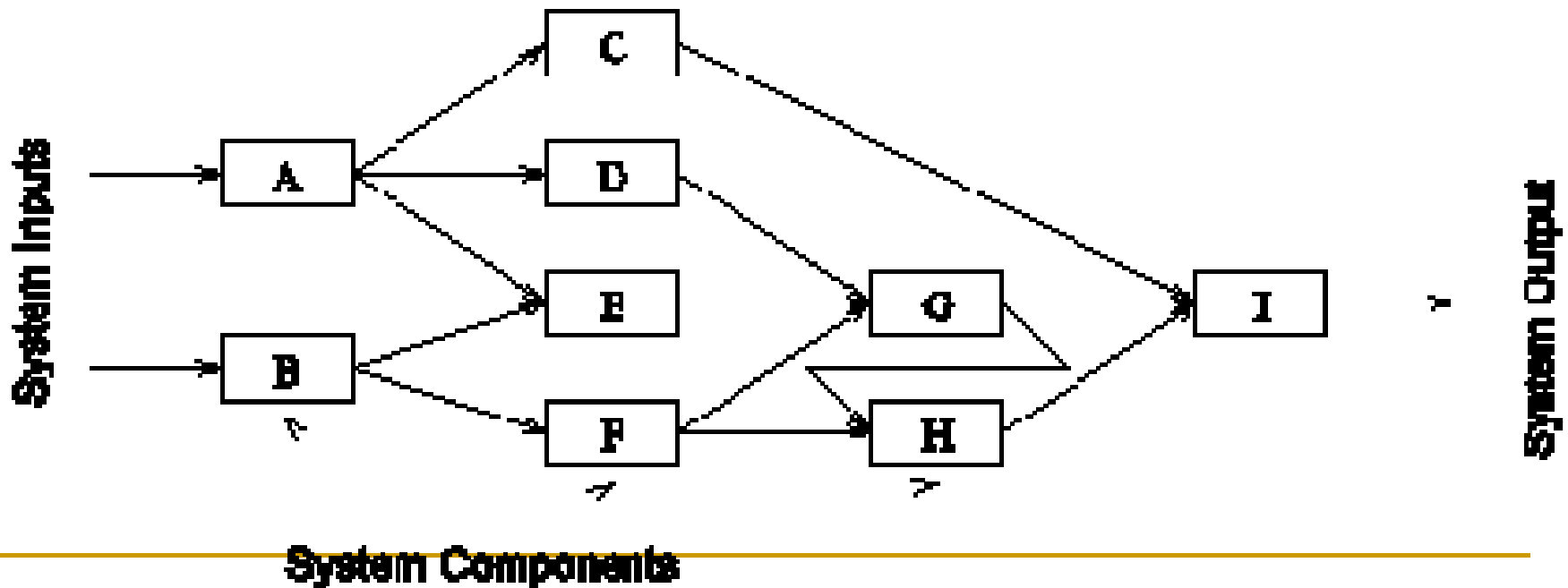
- V niektorých aplikáciách závislosti medzi úlohami nie sú dopredu známe.
 - V takých prípadoch nie je možné dopredu identifikovať úlohy.
 - V takýchto situáciách sú možné dva prístupy
 - konzervatívny, ktorý identifikuje nezávislé úlohy iba ak je zaručené, že medzi nimi nie sú závislosti,
 - optimistický, ktorý naplánuje úlohy aj keď môžu byť chybné.
 - Konzervatívny prístup – slabá súbežnosť.
 - Optimistický prístup – krok späť (Roll-back) ak chyba.
-

Špekulatívna dekompozícia - príklad

- Simulácia pomocou diskretných udalostí (Discrete Event Simulation).
 - Hlavná údajová štruktúra – zoznam udalostí usporiadaných v čase.
 - Udalosti sú spracovávané v poradí danom časom, výsledné udalosti sú umiestňované do zoznamu.
 - Simulácia jedného pracovného dňa
 - Udalosti: Vstať, pripraviť sa, šoferovať do práce, pracovať, obedovať, pracovať, šoferovať späť, večerať, spať.
 - Nezávislé, možnosť spracovať súbežne.
 - Šoferovať do práce – nehoda, žiadna práca sa nekoná.
 - Optimistické plánovanie – „roll-back“ ostatných udalostí.
-

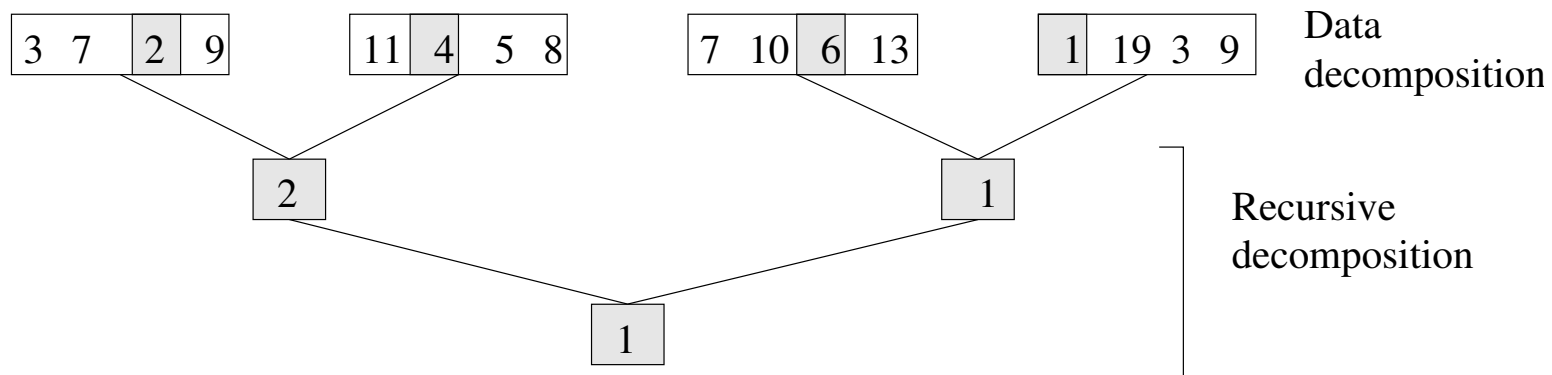
Špekulatívna dekompozícia - príklad

- Simulácia siete uzlov (pásová výroba, počítačová sieť s prechádzajúcimi paketmi)
- Problém – simulácia správania siete pre rôzne vstupy a rôzne oneskorenia na uzloch.



Hybridná dekompozícia

- Často je potrebné vykonať dekompozíciu použitím viacerých spomenutých spôsobov
 - Quicksort – iba rekurzívna dekompozícia obmedzuje súbežnosť, kombinácia rekurzívnej a dátovej dekompozície je vhodnejšia
 - V simulácií pomocou diskretných udalostí môže byť spracovanie udalosti vykonané súbežne – kombinácia špekulatívnej a dátovej dekompozície môže byť vhodná.
 - Hľadanie minimálneho prvku v poli – kombinácia dátovej a rekurzívnej dekompozície:



Charakteristické vlastnosti úloh

- Po dekomponovaní problému na nezávislé úlohy, vlastnosti úloh významne ovplyvňujú výber a výkonnosť paralelného algoritmu.
 - Relevantné charakteristické vlastnosti úloh sú:
 - Spôsob generovania úloh
 - Veľkosť úloh
 - Veľkosť údajov asociovaných s úlohou
-

Spôsob generovania úloh

- Statické generovanie úloh.
 - Súbežné úlohy je možné dopredu identifikovať.
 - Problémy s pravidelnou štruktúrou: násobenie matíc, grafové algoritmy, spracovanie obrazu.
 - Dynamické generovanie úloh
 - Úlohy sú generované v priebehu vykonávania výpočtu.
 - Hranie hier (15 Puzzle).
 - Exploratívna alebo špekulatívna dekompozícia.
-

Veľkosť úloh

- Rovnomerná veľkosť úloh (úlohy rovnakej veľkosti) alebo nerovnomerná veľkosť úloh.
 - Úlohy nerovnomernej veľkosti – možnosť odhadnúť dopredu veľkosť alebo aj nie.
 - Diskrétna optimalizácia – ťažké určiť efektívnu veľkosť stavového priestoru.
-

Veľkosť údajov asociovaných s úlohami

- Veľkosť asociovaných dát môže byť veľká alebo malá v kontexte veľkosti úlohy.
 - Malý kontext úlohy znamená, že algoritmus môže ľahko komunikovať úlohu inému procesu dynamicky (15 Puzzle)
 - Rozsiahly kontext pripútava úlohu k procesu.
-

Charakteristické vlastnosti interakcie medzi úlohami

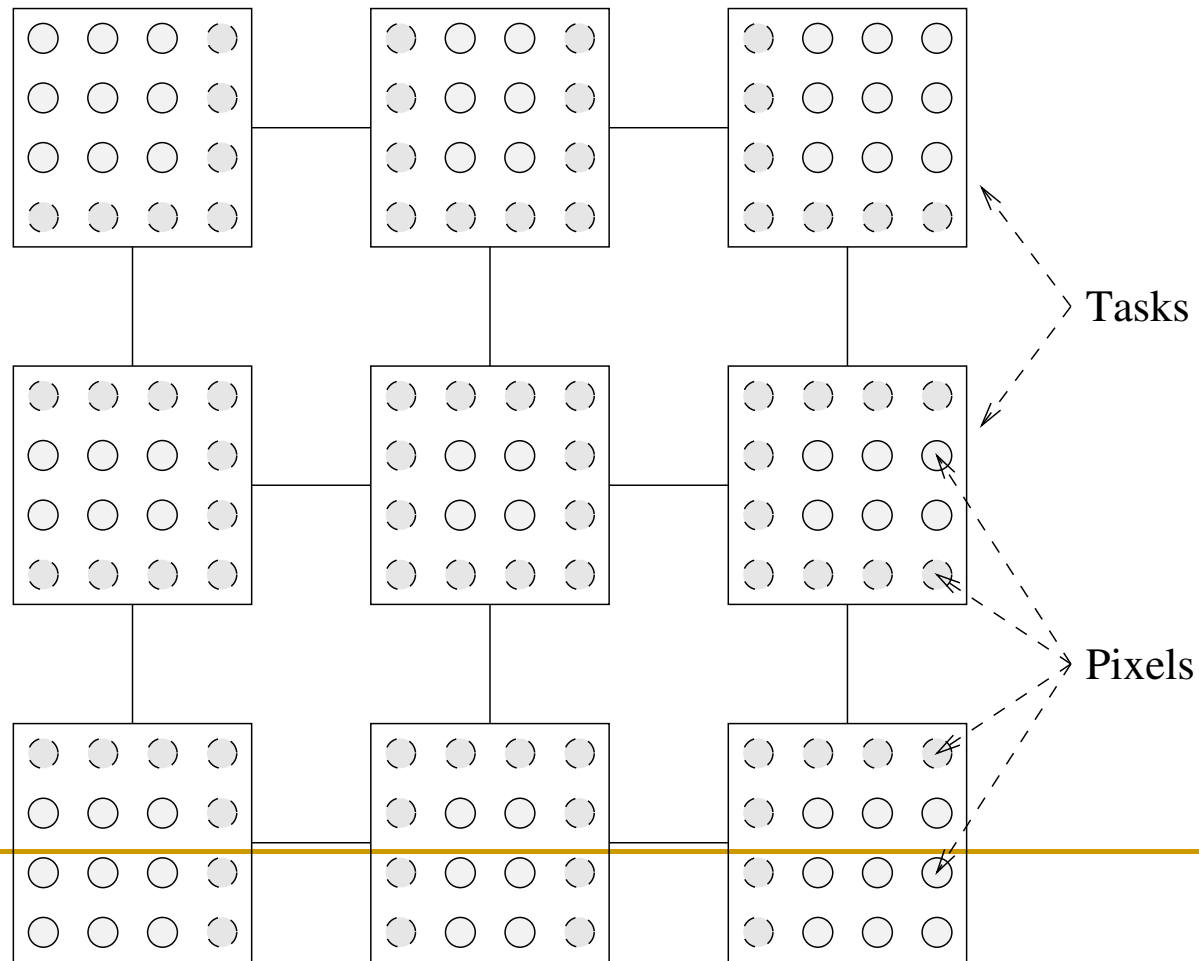
- Úlohy môžu medzi sebou komunikovať rôznymi spôsobmi.
 - Statické interakcie:
 - Úlohy a ich interakcie sú dopredu známe.
 - Zvyčajne jednoduchšie na implementáciu.
 - Dynamické interakcie:
 - Časovanie a interakcie úloh nie je možné dopredu definovať.
 - Zložitejšia implementácia, obzvlášť pri modeloch zasielania správ.
-

Charakteristické vlastnosti interakcie medzi úlohami

- Pravidelné regulárne interakcie:
 - Definovaný vzor v grafe interakcií.
 - Možnosť využiť vzor pre efektívnu implementáciu.
 - Nepravidelné iregulárne interakcie:
 - Interakcie nevykazujú dobre definované typológie v grafe.
-

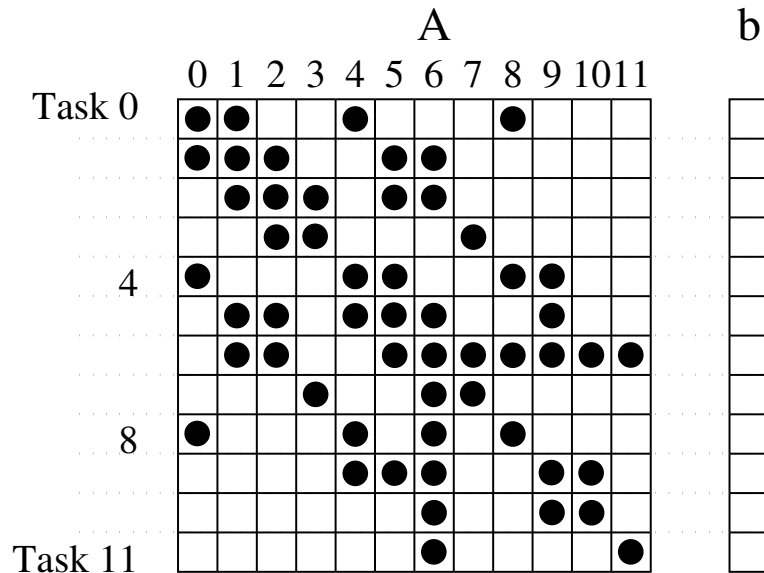
Charakteristické vlastnosti interakcie medzi úlohami - príklad

- Filtrovanie obrazu – komunikácia v 2D mriežke

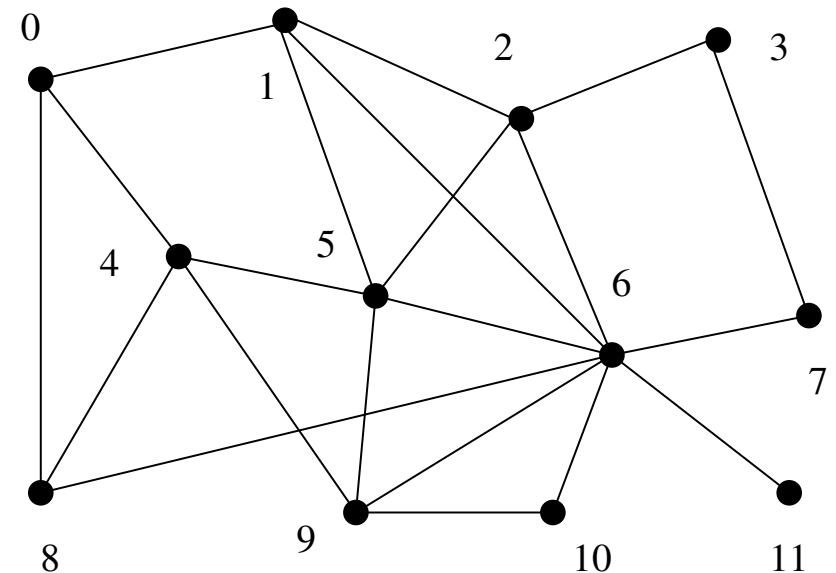


Charakteristické vlastnosti interakcie medzi úlohami - príklad

- Násobenie riedkej matice s vektorom – statický iregulárny vzor



(a)



(b)

Charakteristické vlastnosti interakcie medzi úlohami

- Interakcie iba čítanie alebo zápis a čítanie.
 - Interakcie typu „iba čítanie“ – úlohy iba čítajú údaje prislúchajúce iným úlohám.
 - Interakcie typu „čítanie a zápis“ – úlohy čítajú ale aj modifikujú údaje prislúchajúce iným úlohám.
 - Vo všeobecnosti interakcie typu „čítanie a zápis“ sa ťažšie implementujú, vyžadujú použitie dodatočných synchronizačných primitív.
-

Charakteristické vlastnosti interakcie medzi úlohami

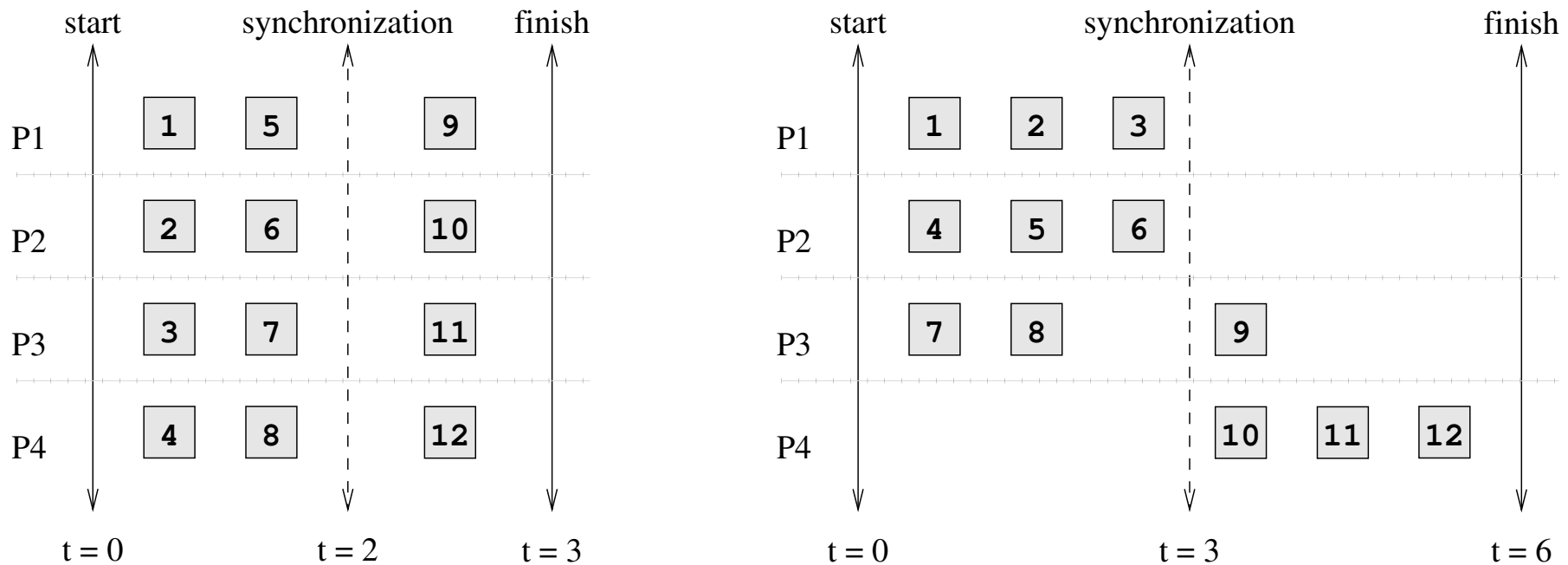
- Interakcie môžu byť „jednostranné“ alebo „obojstranné“.
 - Jednostranné interakcie – iniciované a vykonané iba jednou z dvoch interagujúcich úloh.
 - Obojstranné interakcie – vyžadujú participáciu oboch úloh.
 - Jednostranné interakcie sa ťažšie implementujú v modeloch zasielania správ.
-

Metódy mapovania úloh na procesy

- Po dekomponovaní problému na súbežné úlohy je potrebné úlohy namapovať na procesy vykonateľné na paralelnej platforme
 - Mapovanie musí minimalizovať réžiu (overhead), hlavne
 - Komunikačná réžia
 - Réžia súvisiaca s nečinnosťou procesorov
 - Minimalizácia réžie – často protichodné ciele
 - Napr. triviálna minimalizácia komunikácie mapovaním celého problému na jeden procesor spôsobí veľkú réžiu súvisiacu s nečinnosťou ostatných dostupných procesorov
-

Metódy mapovania na minimalizáciu nečinnosti

- Mapovanie musí súčasne minimalizovať nečinnosť a zároveň zabezpečiť vyvážené zaťaženie procesorov
- Čisté vyvažovanie záťaže nemusí minimalizovať nečinnosť.



(a)

(b)

Metódy mapovania na minimalizáciu nečinnosti

- Metódy mapovania môžu byť statické alebo dynamické:
 - Statické mapovanie
 - Úlohy „vopred“ mapované na procesy
 - Potrebný dobrý odhad veľkosti úloh
 - Problém mapovania môže byť NP úplný
 - Dynamické mapovanie
 - Úlohy mapované na procesy „za behu“
 - Napr. z dôvodu že sú generované „za behu“ alebo ich veľkosť nie je dopredu známa
 - Ostatné faktory determinujúce výber metódy
 - Veľkosť údajov asociovaných s úlohou
 - Charakter problémovej domény
-

Schémy statického mapovania

- Mapovanie založené na dátovej dekompozícií
 - Mapovanie založené na dekomponovaní grafu
 - Hybridné mapovanie
-

Mapovanie založené na dátovej dekompozícii

- Dátovej dekompozícia (vlastník údajov realizuje výpočet nad údajmi) – rozdelenie na podúlohy
- Najjednoduchšia schéma dátovej dekompozície pre matice je 1D bloková schéma rozdelenia

row-wise distribution

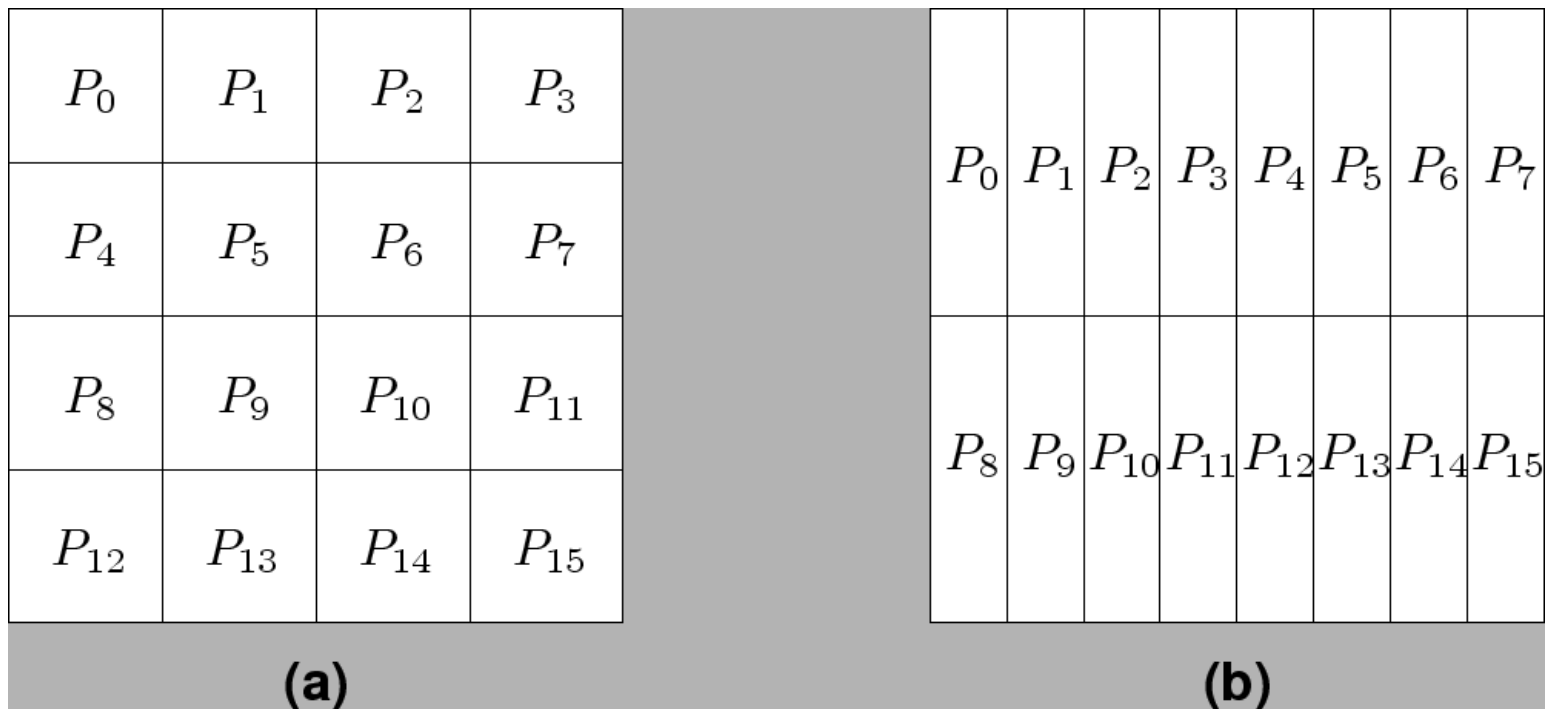
P_0
P_1
P_2
P_3
P_4
P_5
P_6
P_7

column-wise distribution

P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7
-------	-------	-------	-------	-------	-------	-------	-------

Schémy rozdelenia do poľa blokov

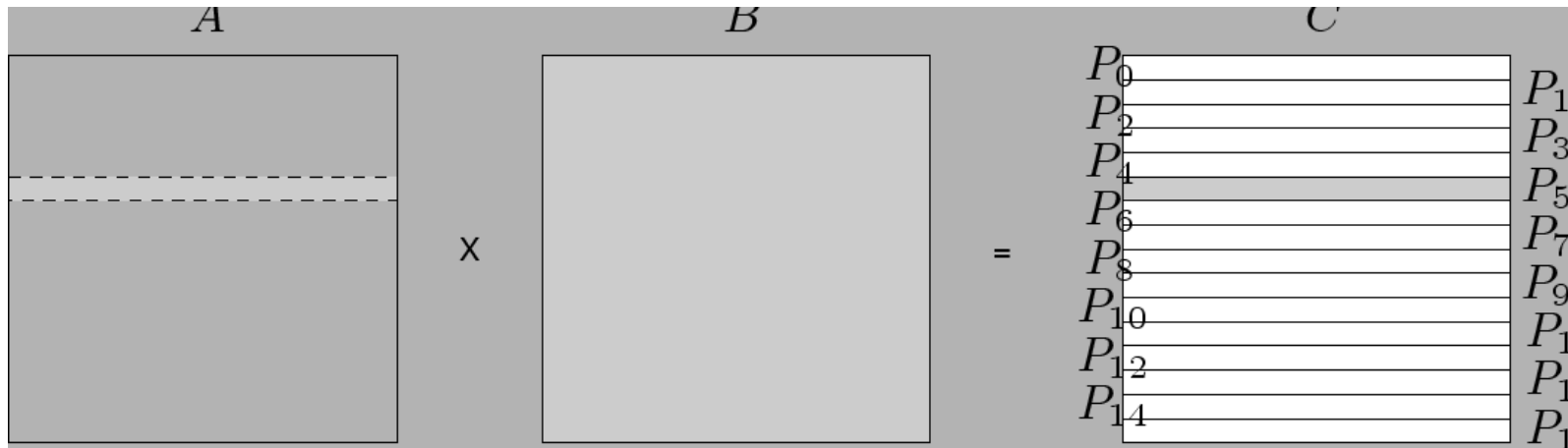
- Možnosť zovšeobecnenia na vyššie dimenzie



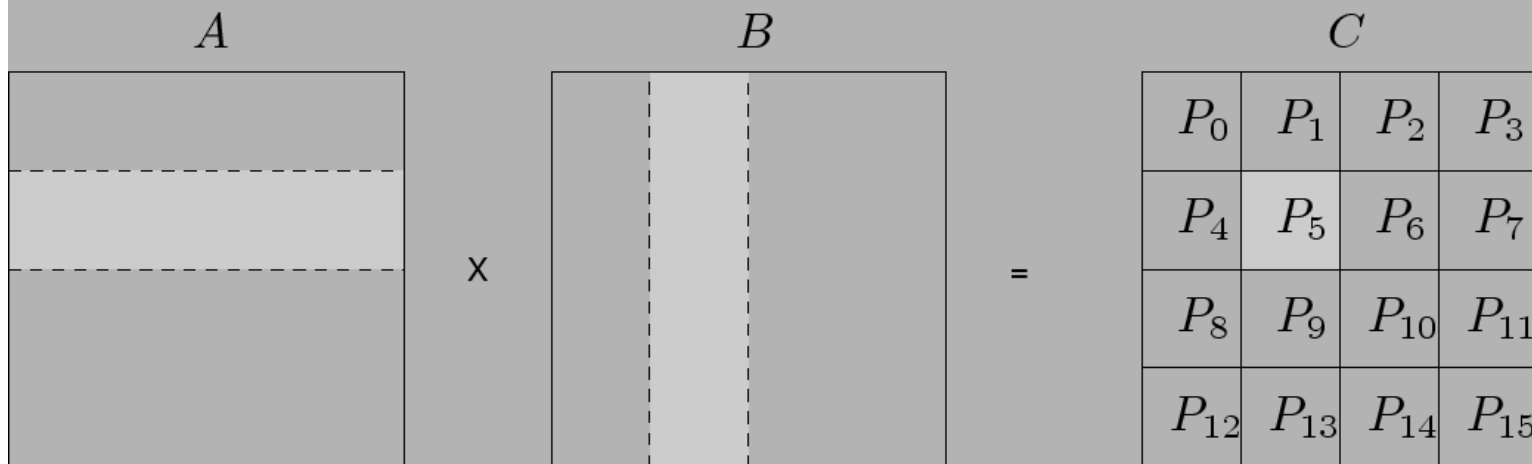
Schémy rozdelenia do poľa blokov - príklad

- Násobenie matíc **A** a **B**, rozdelenie výstupnej matice **C** pomocou dekompozície do blokov
 - Vyváženie záťaže – každá úloha rovnaký počet prvkov výstupnej matice **C** (jeden prvok **C** – jeden skalárny súčin)
 - Výber presnej dekompozície (1-D or 2-D) závisí od zodpovedajúcej komunikačnej réžie
 - Vo všeobecnosti vyššie dimenzie dekompozície umožňujú využiť väčší počet procesorov
-

Zdieľanie dát v úlohe násobenia matic



(a)



(b)

Cyklická a blokovo cyklická dekompozícia

- Ak sa množstvo výpočtu asociovaného s dátovými položkami mení – bloková dekompozícia môže viesť k značnej nerovnováhe z hľadiska záťaže procesorov
 - Jednoduchý príklad – LU dekompozícia
-

LU dekompozícia (LU faktorizácia)

- LU faktorizácia matice – dekompozícia na 14 úloh – nevyrovnaná záťaž

$$\begin{pmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} \succ \begin{pmatrix} L_{1,1} & 0 & 0 \\ L_{2,1} & L_{2,2} & 0 \\ L_{3,1} & L_{3,2} & L_{3,3} \end{pmatrix} \cdot \begin{pmatrix} U_{1,1} & U_{1,2} & U_{1,3} \\ 0 & U_{2,2} & U_{2,3} \\ 0 & 0 & U_{3,3} \end{pmatrix}$$

$$1: A_{1,1} \rightarrow L_{1,1}U_{1,1}$$

$$2: L_{2,1} = A_{2,1}U_{1,1}^{-1}$$

$$3: L_{3,1} = A_{3,1}U_{1,1}^{-1}$$

$$4: U_{1,2} = L_{1,1}^{-1}A_{1,2}$$

$$5: U_{1,3} = L_{1,1}^{-1}A_{1,3}$$

$$6: A_{2,2} = A_{2,2} - L_{2,1}U_{1,2}$$

$$7: A_{3,2} = A_{3,2} - L_{3,1}U_{1,2}$$

$$8: A_{2,3} = A_{2,3} - L_{2,1}U_{1,3}$$

$$9: A_{3,3} = A_{3,3} - L_{3,1}U_{1,3}$$

$$10: A_{2,2} \rightarrow L_{2,2}U_{2,2}$$

$$11: L_{3,2} = A_{3,2}U_{2,2}^{-1}$$

$$12: U_{2,3} = L_{2,2}^{-1}A_{2,3}$$

$$13: A_{3,3} = A_{3,3} - L_{3,2}U_{2,3}$$

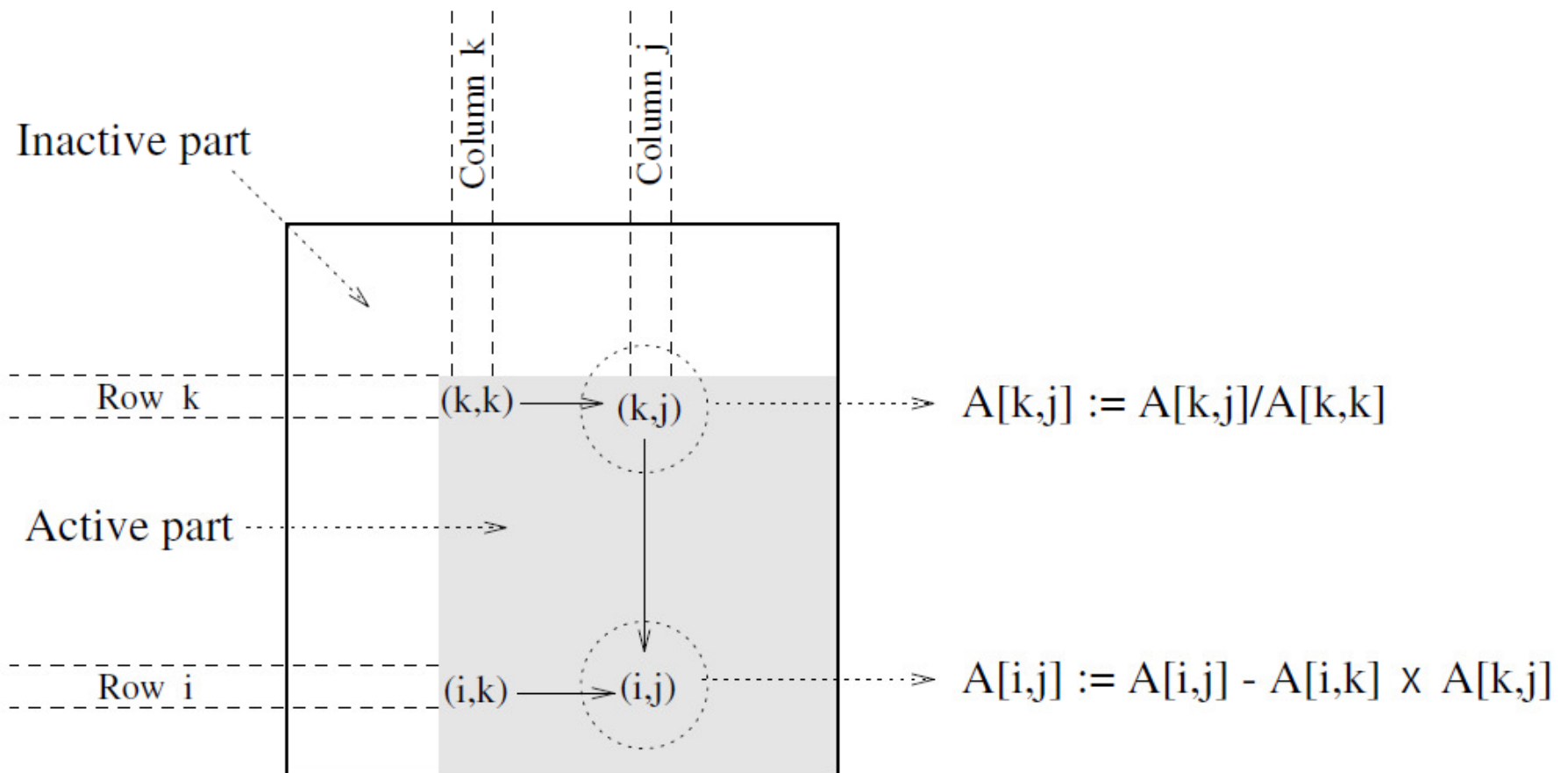
$$14: A_{3,3} \rightarrow L_{3,3}U_{3,3}$$

Blokovo cyklická dekompozícia

- Variácia mapovania pri blokovo cyklickej dekompozícii môže byť použitá na odľahčenie nerovnomernosti zaťaženia procesorov a ich lepšie využitie
 - Rozdelenie poľa dát počet blokov >> počet procesorov
 - Bloky sú priradené procesorom jeden po druhom (Round Robin) tak, že každý proces obdrží niekoľko nesusediacich blokov
-

Blokovo cyklická dekompozícia

- Gaussova eliminácia – aktívna oblasť sa mení



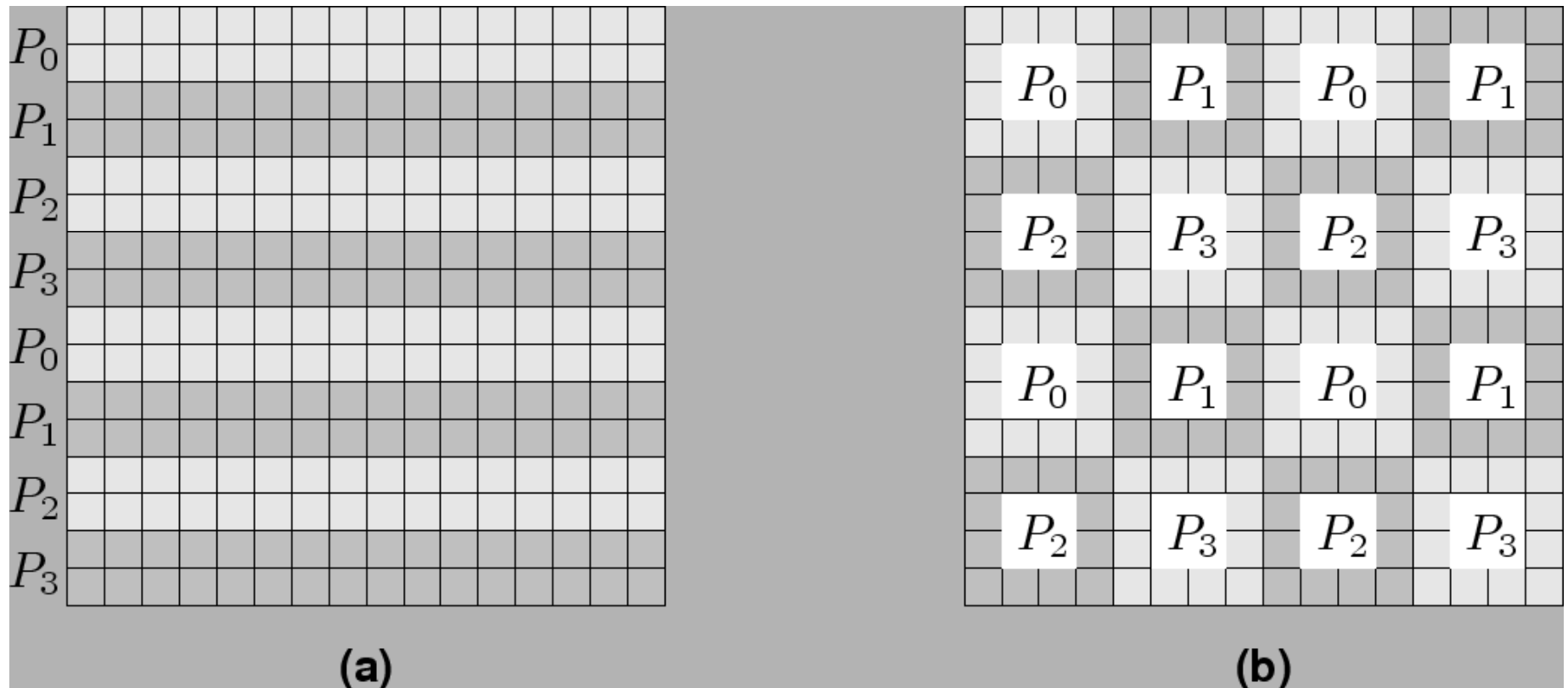
Blokovo cyklická dekompozícia – príklad naivnej dekompozície

- Naivné mapovanie úloh na procesy (dátová dekompozícia)

P₀ T ₁	P₃ T ₄	P₆ T ₅
P₁ T ₂	P₄ T ₆ T ₁₀	P₇ T ₈ T ₁₂
P₂ T ₃	P₅ T ₇ T ₁₁	P₈ T ₉ T ₁₃ T ₁₄

Blokovo cyklická dekompozícia - príklad

- Jedno a dvoj dimenzionálna dekompozícia na 4 procesy



Blokovo cyklická distribúcia

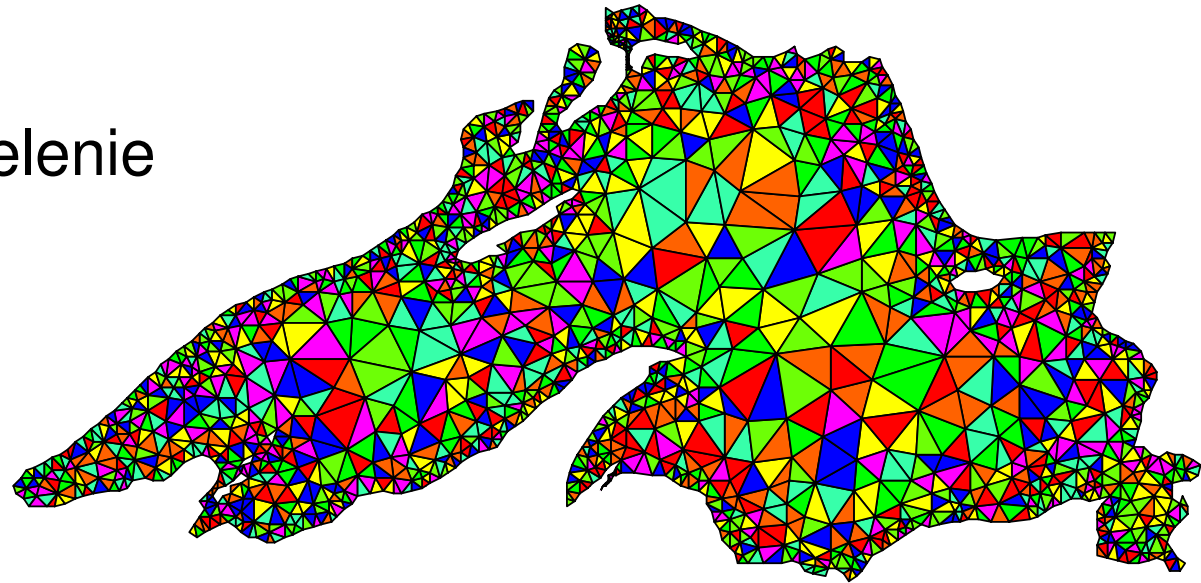
- Cyklická distribúcia je špeciálny prípad s veľkosťou bloku je 1
 - Blokovaná distribúcia je špeciálny prípad s veľkosťou bloku n/p , kde n je dimenzia matice a p je počet procesorov
-

Dátová dekompozícia založená na rozdelení grafu

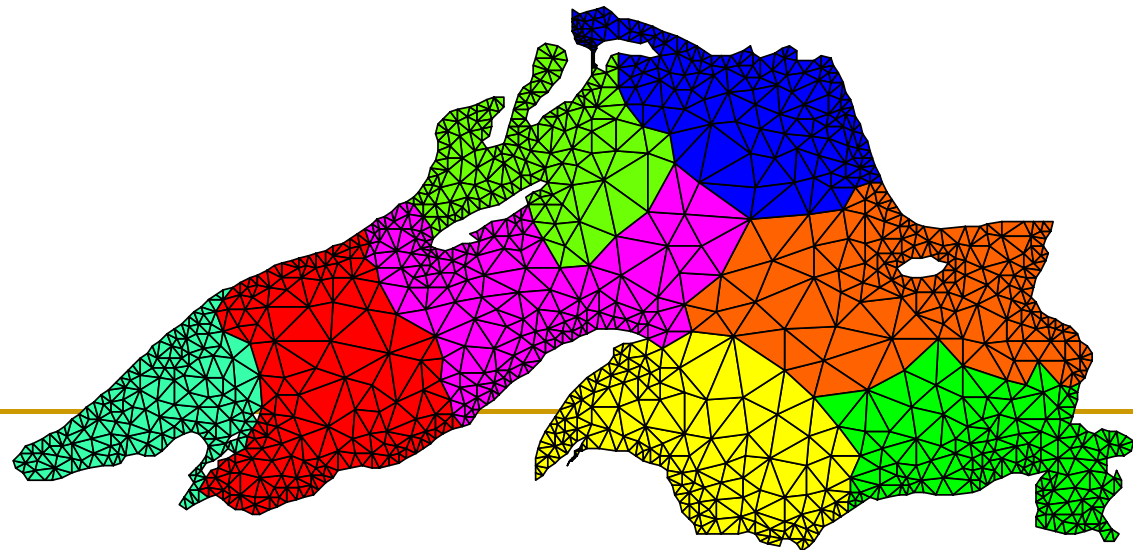
- Riedke matice – bloková dekompozícia je náročná
 - Násobenie riedkej matice vektorom
 - Graf matice je užitočným indikátorom množstva práce (počet uzlov) a komunikácie (stupeň vrcholov)
 - Rovnomerné rozdelenie grafu medzi procesy vzhľadom na počet uzlov, pričom sa minimalizuje počet hrán medzi partíciami grafu
-

Rozdelenie grafu Horného jazera

- Náhodné rozdelenie



- Rozdelenie minimalizujúce počet hrán medzi partíciami

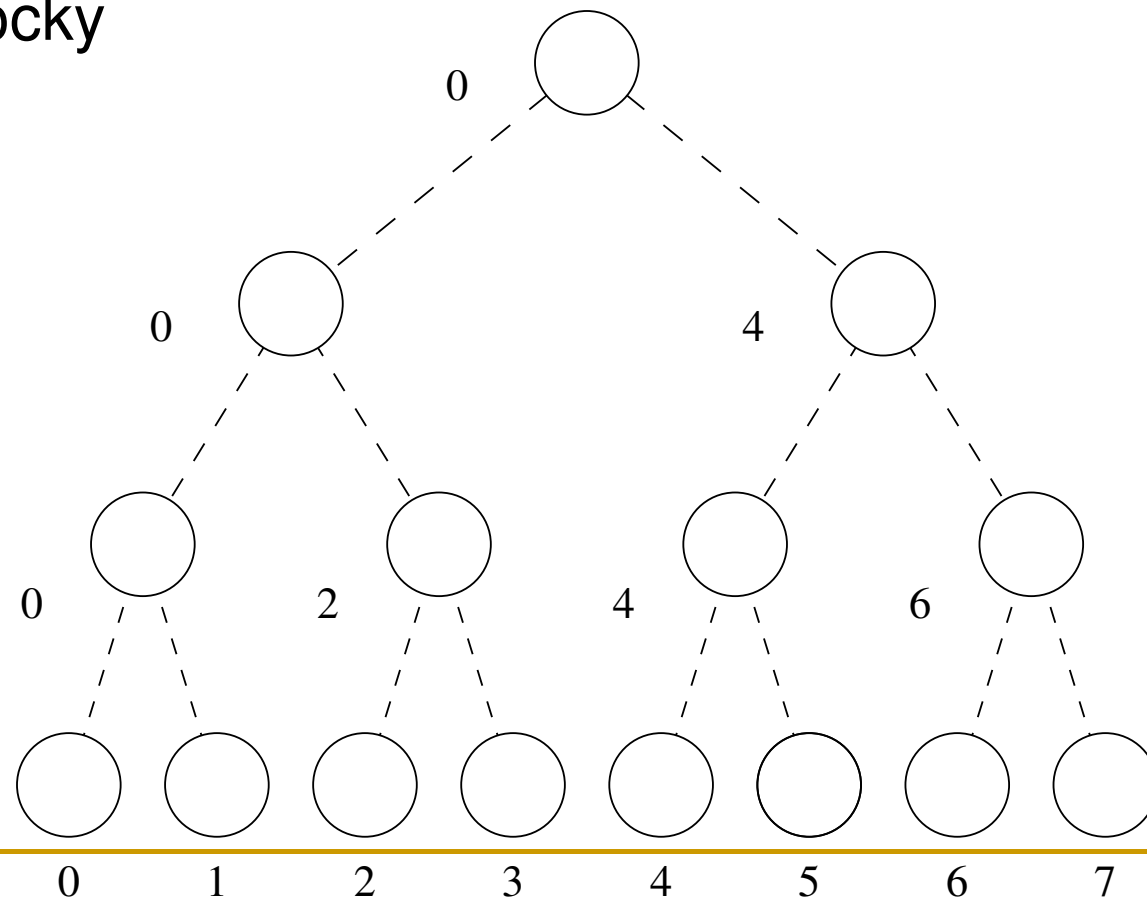


Mapovanie založené na dekompozícií na úlohy

- Rozdelenie daného grafu závislostí medzi úlohami na procesy
 - Určenie optimálneho mapovania pre všeobecný graf závislostí je NP-úplný problém
 - Existujú kvalitné heuristiky pre štruktúrované grafy
-

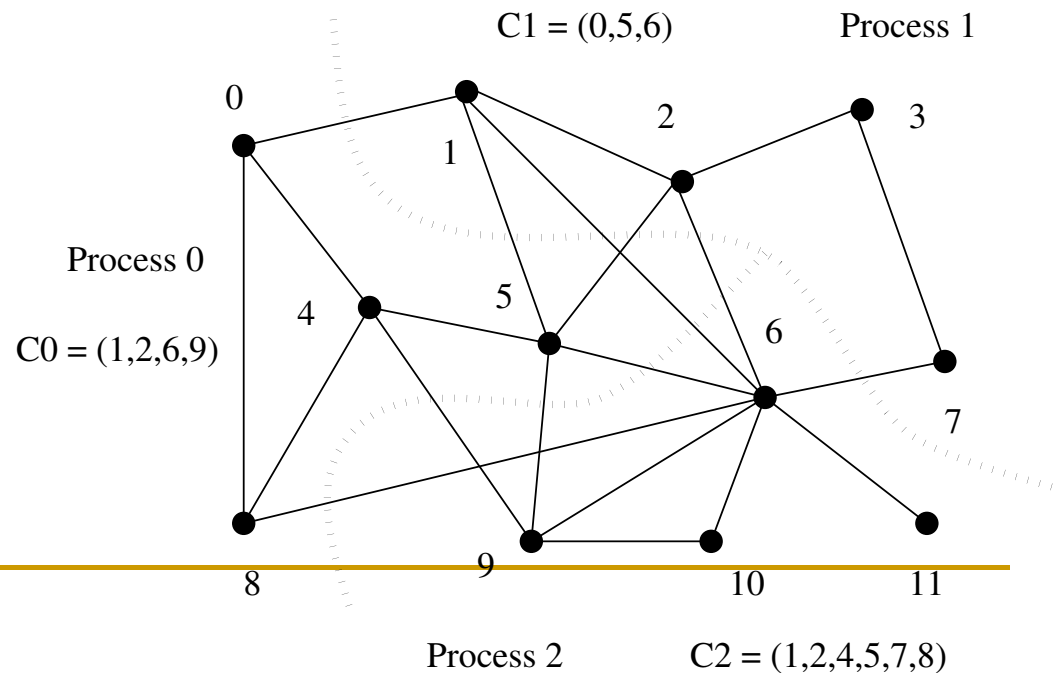
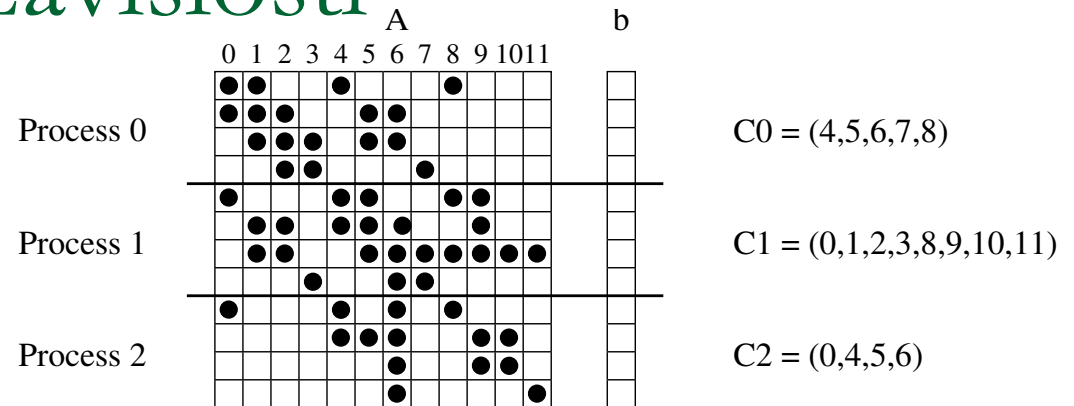
Dekompozícia na úlohy: mapovanie binárneho stromu (grafu závislostí)

- Graf závislostí algoritmu „quick-sort“ na procesy hyperkocky



Dekompozícia na úlohy: mapovanie riedkeho grafu závislostí

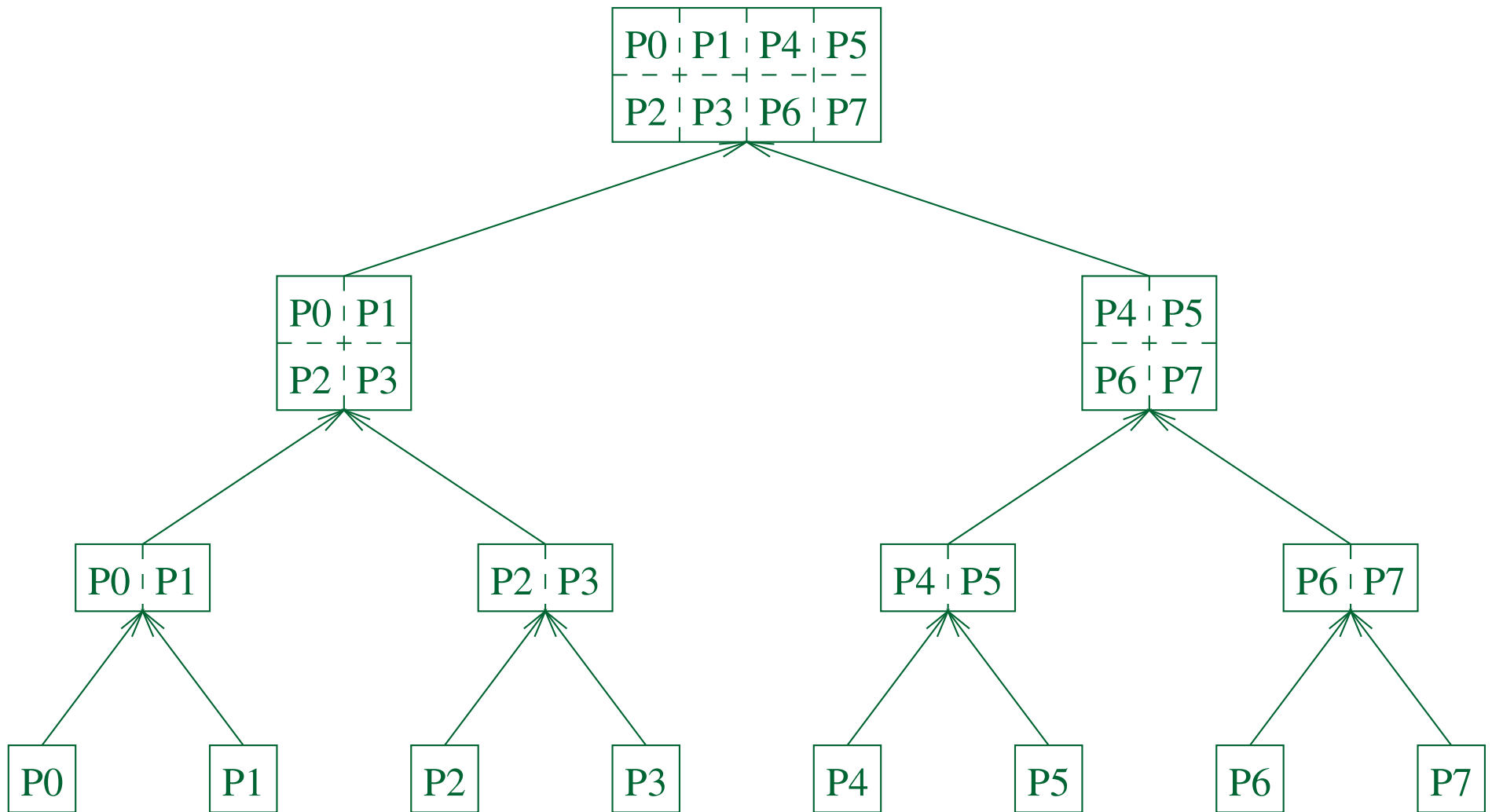
- Graf komunikácie násobenia riedkej matice a vektora



Hierarchické mapovanie

- Jediné mapovanie môže byť neadekvátne pre niektoré úlohy
 - Napr. mapovanie úloh binárneho stromu (alg. „quick-sort“) na veľký počet procesorov nie je možné
 - Mapovanie na základe dekompozície na úlohy na hornej úrovni môže byť kombinované s dátovou dekompozíciou na každej úrovni
-

Hierarchické mapovanie



Schémy dynamického mapovania

- Dynamické mapovanie niekedy nazývané tiež dynamické vyrovňovanie záťaže, nakoľko vyrovňovanie záťaže je hlavnou motiváciou pre dynamické mapovanie
 - Schémy dynamického mapovania môžu byť centralizované alebo decentralizované
-

Centralizované dynamické mapovanie

- Procesy sú označené ako riadiaci proces a podriadené procesy („Master“ a „Slaves“)
 - Ak proces nemá prácu, vyžiada si ju od riadiaceho procesu
 - S rastúcim počtom procesov komunikácia s riadiacim procesom môže byť úzkym hrdlom
 - Proces si môže „vzdvihnúť“ skupinu úloh (chunk) – Chunk Sheduling
 - Príliš veľké skupiny – nerovnováha
 - Viaceré schémy – postupné zmenšovanie veľkosti skupiny
-

Distribúované dynamické mapovanie

- Každý proces môže posielat' alebo prijímať prácu od iných procesov
 - Zmierňuje to úzke hrdlo súvisiace s centralizovanou schémou dynamického mapovania
 - Štyri kritické otázky:
 - Ako sú odosielateľ a prijímateľ spárované
 - Kto iniciuje presun práce
 - Ako veľa práce je prenesenej
 - Kedy je presun odštartovaný
 - Závisí od doménovej oblasti
-

Minimalizácia réžie súvisiacej s interakciami medzi úlohami

- Maximalizácia dátovej lokálnosti
 - Znovupoužitie údajov, reštrukturalizácia výpočtu tak, aby údaje mohli byť znovu použité v kratšom časovom okne
 - Minimalizácia objemu prenášaných údajov
 - S každým presunutým slovom je spojená réžia a tak je potrebné minimalizovať objem prenášaných údajov
 - Minimalizácia frekvencie interakcií
 - S inicializáciou každej interakcie je spojená réžia
 - Je potrebné zoskupovať viaceré interakcie do jednej
 - Minimalizácia súťaženia a úzkych hrdiel
 - Použitie decentralizovaných metód
 - Replikácia údajov, kde je to potrebné
-

Minimalizácia réžie súvisiacej s interakciami medzi úlohami

- Prelínanie výpočtov a interakcií, zakrývanie latencie:
 - Používanie neblokujúcej komunikácie
 - Viacvláknovosť
 - Skoré načítavanie
 - Replikácia údajov alebo výpočtov
 - Používanie skupinovej komunikácie (nie bod – bod komunikácie)
 - Prelínanie interakcií s inými interakciami
-

Modely paralelných algoritmov

- Model algoritmu
 - Spôsob štruktúrovania paralelného algoritmu výberom metódy dekompozície a mapovania a aplikáciu vhodnej stratégie na minimalizáciu interakcií
 - Dátovo paralelný model
 - Úlohy sú staticky mapované na procesy a každá úloha vykonáva podobné výpočty nad rôznymi údajmi
 - Model grafu úloh
 - Počnúc grafom závislosti medzi úlohami vzťahy medzi úlohami sú využité na podporu lokálnosti a redukciu režie spojenej s interakciou
-

Modely paralelných algoritmov

- Model „nadriadený – podriadení“
 - Jeden alebo viac procesov generujú prácu a alokujú ju na podriadené procesy, statická alebo dynamická alokácia
 - Prúdová linka / producenti - konzumenti
 - Prúd údajov je prenášaný postupnosťou procesov, každý vykoná nad údajmi danú úlohu
 - Hybridné modely
 - Kompozícia viacerých modelov aplikovaných hierarchicky
 - Viaceré modely aplikované sekvenčne počas rôznych fáz paralelného algoritmu
-

Zdroje

- Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar. Introduction to Parallel Computing, 2nd Edition, Addison-Wesley 2003, „Introduction to Parallel Computing“ <http://www-users.cs.umn.edu/~karypis/parbook/>
 - Obrázky prevzaté z:
 - [Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar. Introduction to Parallel Computing, 2nd Edition, Addison-Wesley 2003, „Introduction to Parallel Computing“ http://www-users.cs.umn.edu/~karypis/parbook/](http://www-users.cs.umn.edu/~karypis/parbook/)
-