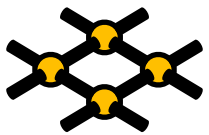
Obecnější aplikované úvahy o paralelizaci část II

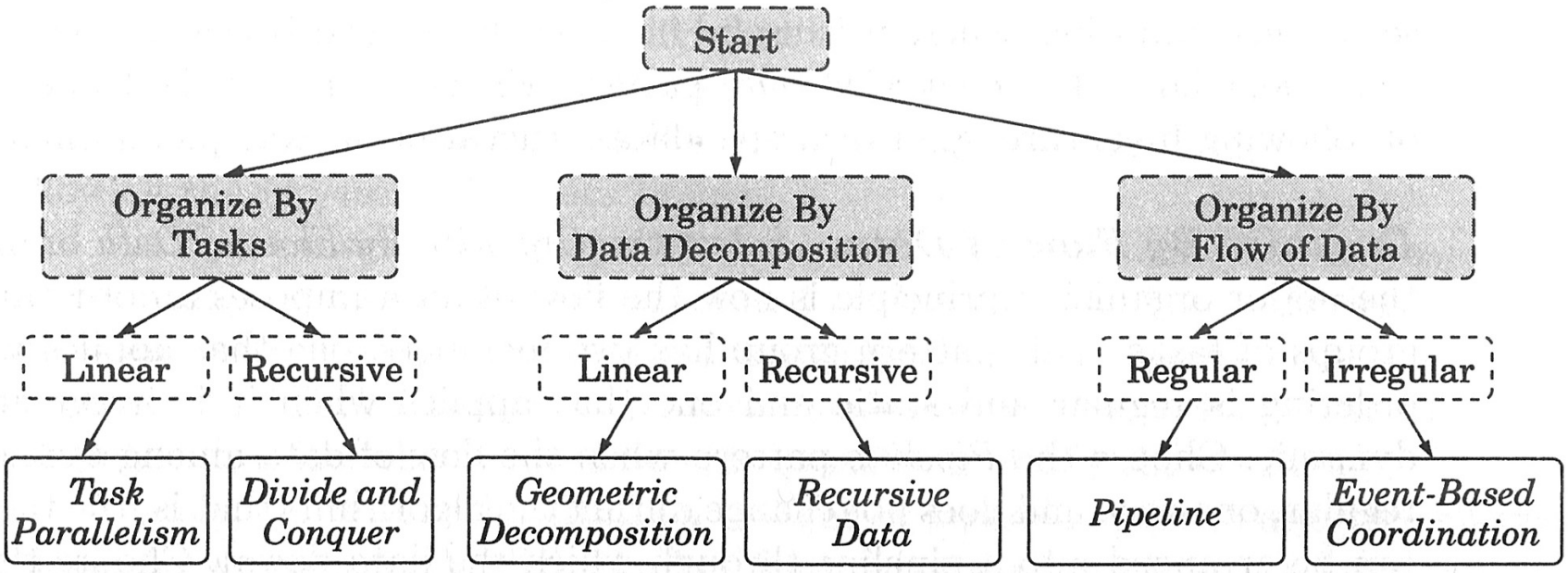
Petr Holub
hopet@ics.muni.cz



Laboratoř pokročilých síťových technologií

PV197
2009–12–14

Volba vzorů


Start
Organize By Tasks
Organize By Data Decomposition
Organize By Flow of Data
Linear
Recursive
Linear
Recursive
Regular
Irregular
Task Parallelism
Divide and Conquer
Geometric Decomposition
Recursive Data
Pipeline
Event-Based Coordination

Paralelismus úloh

- Dekompozice podle úloh
  - potřeba vygenerovat dostatečné množství úloh
- Řešení závislostí:
  - odstranitelné závislosti

```
1 int ii=0, jj=0;

3 for (int i=0; i<N; i++) {
        ii = ii+1;
5       d[ii] = dlouhy_vypocet(ii);
        jj = jj+i;
7       a[jj] = jiny_dlouhy_vypocet(jj);
}
```

```
1 for (int i=0; i<N; i++) {
        d[i] = dlouhy_vypocet(i);
3       a[i*(i+1)/2] = jiny_dlouhy_vypocet(i*(i+1)/2);
}
```

  - rozdělitelné závislosti: lokální výpočet + redukce
  - jiné typy závislostí: řešení sdílením dat

Paralelismus úloh

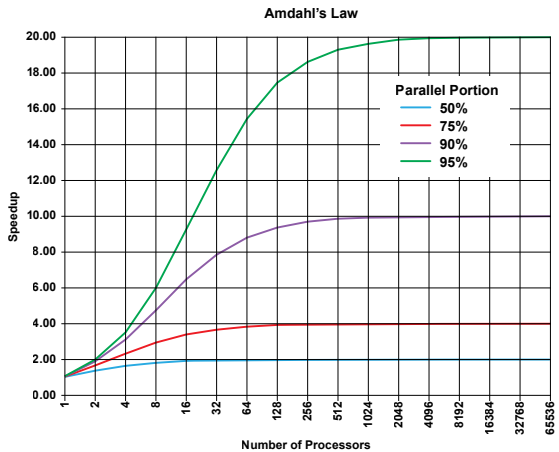
- Plánování úloh:
  - statické: vygenerování úloh na počátku (např. paralelizace smyček předem dané velikosti)
  - dynamické
    - dynamické generování úloh do centrální fornty
    - work stealing
- Idiomatické typy:
  - embarrassingly parallel tasks
  - úlohy nad replikovanými daty (1. vykopírování dat do lokálních proměnných, 2. výpočet, 3. seskládání výsledků)
- Příklady:
  - zpracování obrazu: pokud je zpracování pro oblasti nezávislé na okolí
  - raytracing, PET: dekompozice na jednotlivé paprsky
  - molekulová dynamika: dekompozice na jednotlivé atomy při výpočtu příspěvků sil; dvouelektronové integrály u kvantových metod

Rozděl a panuj

- Klasický přístup pro rekurzivní úlohy
- Problém se sekvenční režií dělení na podúlohy

$$S(P) = \frac{T_{tot}(1)}{T_{tot}(P)} = \frac{1}{\gamma + \frac{1-\gamma}{P}}$$

kde $\gamma = \frac{T_{serial}}{T_{tot}(1)}$.



Zdroj: http://en.wikipedia.org/wiki/File:AmdahlsLaw.svg

- Příklady: mergesort, diagonalizace matic (hledání vlastního vektoru a vlastních hodnot – hledání matice Q takové že $Q^T \cdot T \cdot Q$ je diagonální), rychlá multipólová metoda (FMM)

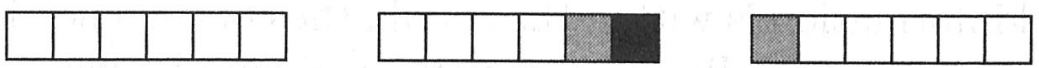
Geometrická dekompozice

- Hledání geometrického dělení zpracovávaných dat
  - data dělitelná na velké množství nezávislých segmentů
  - řešení hranic mezi segmenty – ghost boundary
- Příklad – vedení tepla:

$$\frac{\partial U}{\partial t} = \alpha \frac{\partial^2 U}{\partial x^2}$$

```
ukp1[i] = uk[i] + (dt/(dx*dx))*(uk[u+1] + uk[u-1] - 2*uk[i]);
```

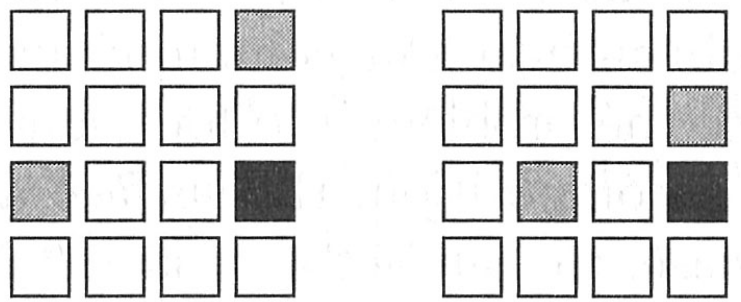
update requiring only local info



update requiring information from another chunk

Geometrická dekompozice

- Příklad – násobení matic:



- Kroky:
  - rozklad dat
    - poměr mezi množstvím vyměňovaných dat a všech dat
    - dělení matice na 4ks: sloupce ($5N/2$) vs. čtverce ($2N$)
  - výměna dat před novým krokem výpočtu
  - výpočet na vlastním segmentu

Geometrická dekompozice

- Rozložení úloh přes procesory:
  - minimalizace režie výměn dat
  - efektivita využití úloh (např. při eliminacích matic mohou procesory hladovět $\Longrightarrow$ cyklické přidělování bloků)

  1D dekompozice:

dekompozice

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$	$a_{0,6}$	$a_{0,7}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$	$a_{1,6}$	$a_{1,7}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$	$a_{2,6}$	$a_{2,7}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$	$a_{3,6}$	$a_{3,7}$
$a_{4,0}$	$a_{4,1}$	$a_{4,2}$	$a_{4,3}$	$a_{4,4}$	$a_{4,5}$	$a_{4,6}$	$a_{4,7}$
$a_{5,0}$	$a_{5,1}$	$a_{5,2}$	$a_{5,3}$	$a_{5,4}$	$a_{5,5}$	$a_{5,6}$	$a_{5,7}$
$a_{6,0}$	$a_{6,1}$	$a_{6,2}$	$a_{6,3}$	$a_{6,4}$	$a_{6,5}$	$a_{6,6}$	$a_{6,7}$
$a_{7,0}$	$a_{7,1}$	$a_{7,2}$	$a_{7,3}$	$a_{7,4}$	$a_{7,5}$	$a_{7,6}$	$a_{7,7}$
$UE(0)$		$UE(1)$		$UE(2)$		$UE(3)$	

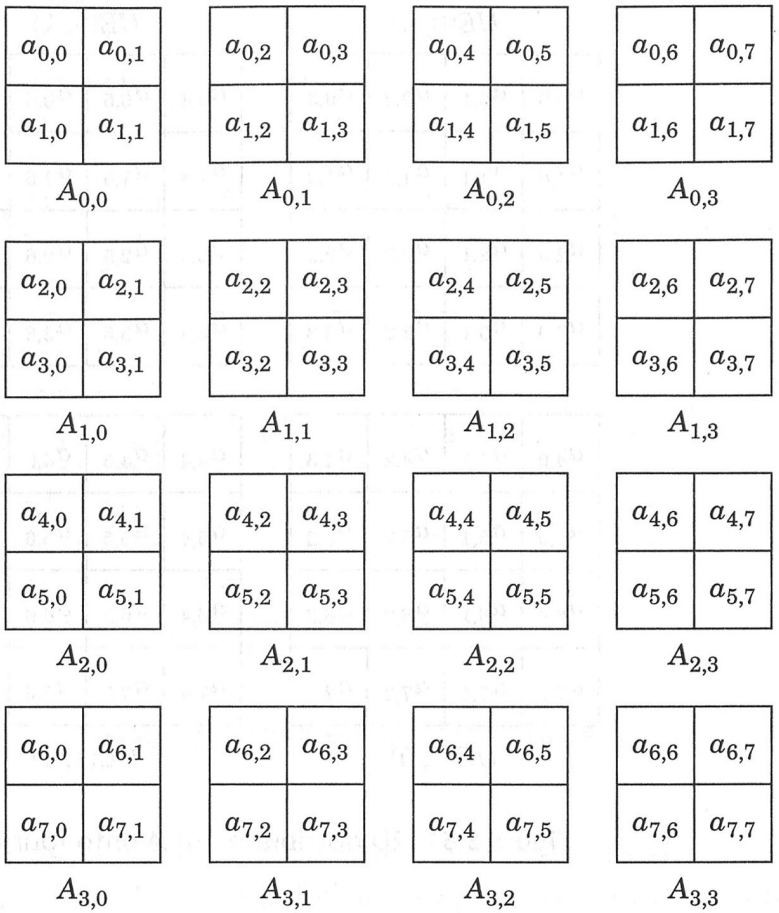
přiřazení ($j \mod 4$)

$a_{0,0}$	$a_{0,4}$	$a_{0,1}$	$a_{0,5}$	$a_{0,2}$	$a_{0,6}$	$a_{0,3}$	$a_{0,7}$
$a_{1,0}$	$a_{1,4}$	$a_{1,1}$	$a_{1,5}$	$a_{1,2}$	$a_{1,6}$	$a_{1,3}$	$a_{1,7}$
$a_{2,0}$	$a_{2,4}$	$a_{2,1}$	$a_{2,5}$	$a_{2,2}$	$a_{2,6}$	$a_{2,3}$	$a_{2,7}$
$a_{3,0}$	$a_{3,4}$	$a_{3,1}$	$a_{3,5}$	$a_{3,2}$	$a_{3,6}$	$a_{3,3}$	$a_{3,7}$
$a_{4,0}$	$a_{4,4}$	$a_{4,1}$	$a_{4,5}$	$a_{4,2}$	$a_{4,6}$	$a_{4,3}$	$a_{4,7}$
$a_{5,0}$	$a_{5,4}$	$a_{5,1}$	$a_{5,5}$	$a_{5,2}$	$a_{5,6}$	$a_{5,3}$	$a_{5,7}$
$a_{6,0}$	$a_{6,4}$	$a_{6,1}$	$a_{6,5}$	$a_{6,2}$	$a_{6,6}$	$a_{6,3}$	$a_{6,7}$
$a_{7,0}$	$a_{7,4}$	$a_{7,1}$	$a_{7,5}$	$a_{7,2}$	$a_{7,6}$	$a_{7,3}$	$a_{7,7}$
$UE(0)$		$UE(1)$		$UE(2)$		$UE(3)$	

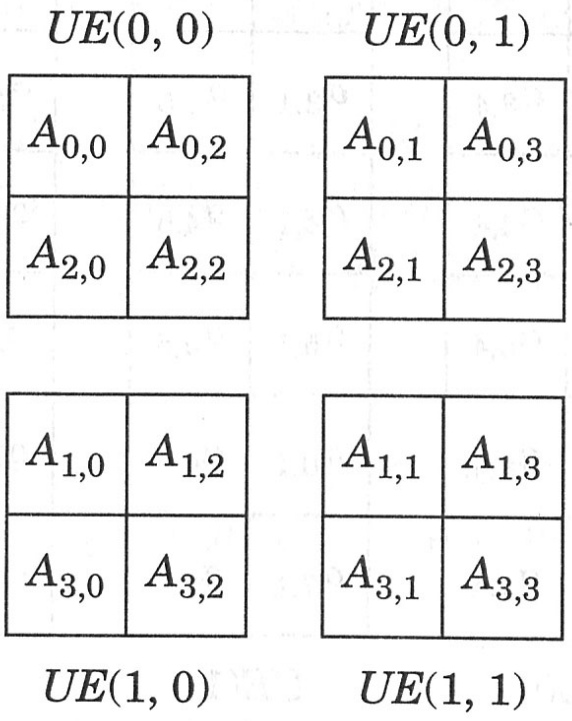
Geometrická dekompozice

- 2D dekompozice:

dekompozice



přiřazení ($i \mod 2, j \mod 2$)



Geometrická dekompozice

- Maskování latence komunikace – překryv komunikace a výpočtu
  1. neblokující volání zahajující komunikaci (distribuci ghost boundary)
  2. výpočet vnitřních prvků
  3. čekání na dokončení komunikace
  4. výpočet hodnot okrajových prvků, závisejících na ghost boundary

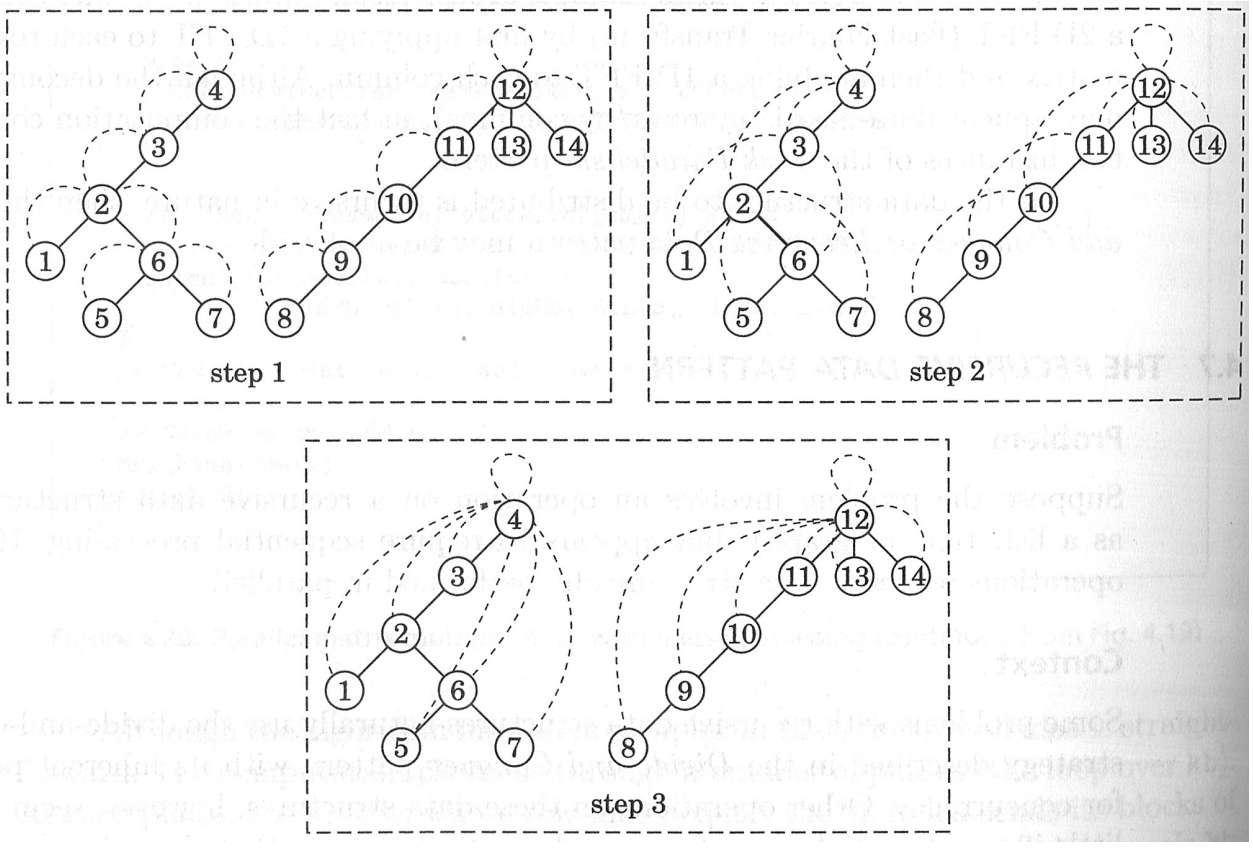
  - Příklady pro MPI:
    `MPI_Isend()`, **`MPI_Irecv()`**
    `MPI_Wait()`

Rekurzivní pohled na data

- Pohled na data, kde nejde přímočaře použít přístup rozděl a panuj z povahy algoritmu
- Příklad: les stromů (každý uzel si drží svého předchůdce, kořen je sám sobě předchůdcem), hledáme pro každý uzel kořen
  - sekvenčně: z každého kořene provedeme DFS – $\mathcal{O}(N)$
- Paralelně:
  - zavedeme si pro každého předchůdce-předchůdce
  - výpočet opakujeme až do konvergence
  - postupujeme na všech N uzlech paralelně
  - celkově $\mathcal{O}(N \log N)$, ovšem při paralelním provedení efektivně jen $\mathcal{O}(\log N)$
- Zhoršení celkové složitosti je pro tuto dekompozici typické
- Potřeba sdílet výsledky po každém kroku (lockstep) (podpora např. v řadě SIMD strojů a High-Performance Fortranu)

Rekurzivní pohled na data

- Kroky:



Rekurzivní pohled na data

- Pohled na data, kde nejde přímočaře použít přístup rozděl a panuj z povahy algoritmu
- Příklad: les stromů (každý uzel si drží svého předchůdce, kořen je sám sobě předchůdcem), hledáme pro každý uzel kořen
  - sekvenčně: z každého kořene provedeme DFS – $\mathcal{O}(N)$
- Paralelně:
  - zavedeme si pro každého předchůdce-předchůdce
  - výpočet opakujeme až do konvergence
  - postupujeme na všech N uzlech paralelně
  - celkově $\mathcal{O}(N \log N)$, ovšem při paralelním provedení efektivně jen $\mathcal{O}(\log N)$
- Zhoršení celkové složitosti je pro tuto dekompozici typické
- Potřeba sdílet výsledky po každém kroku (lockstep) (podpora např. v řadě SIMD strojů a High-Performance Fortranu)

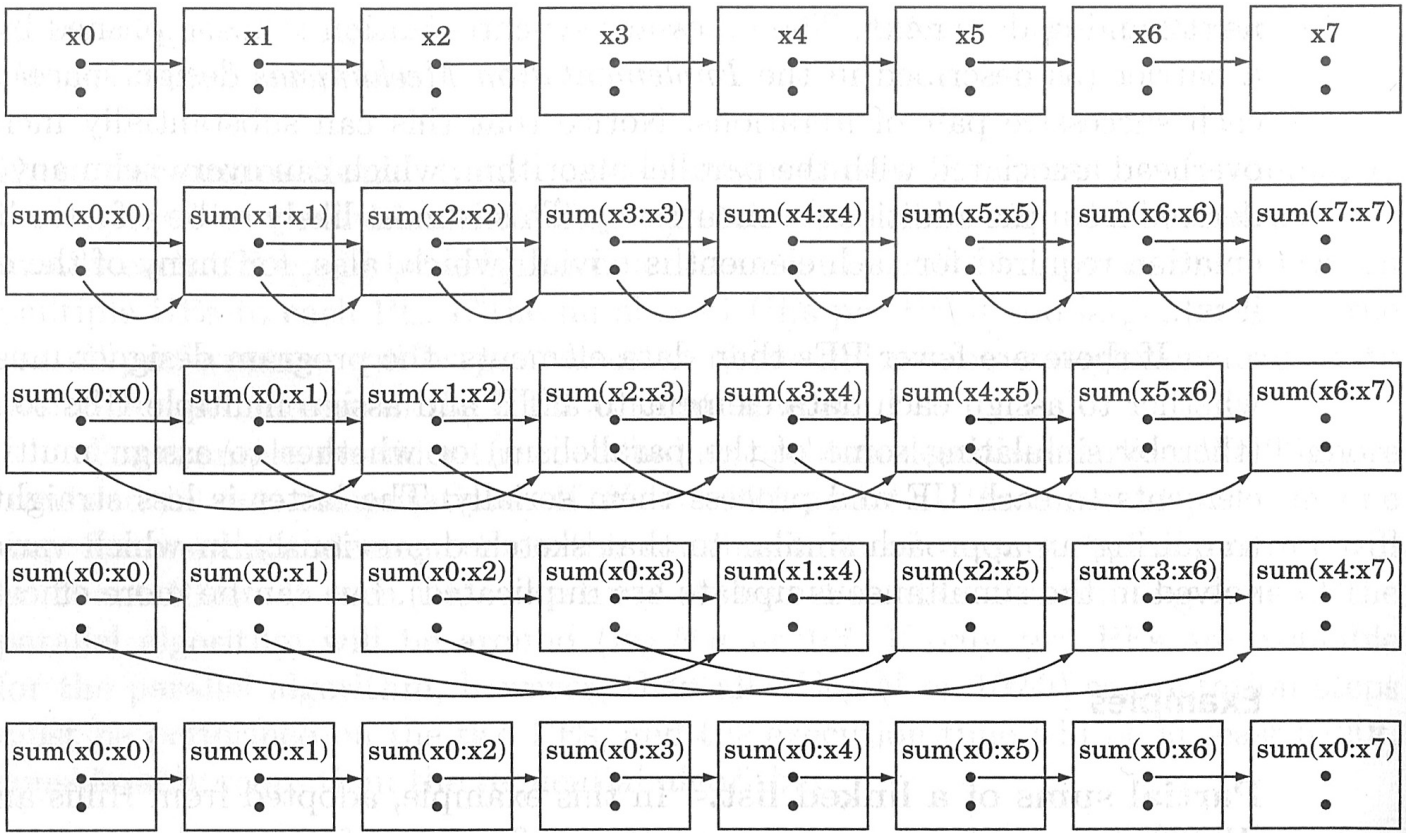
Rekurzivní pohled na data

- Součty prefixů
  - potřebujeme sečíst pro *n*-tý prvek všech 1 až *n* předchůdců
  - opět rekurzivní zdvojování

```
1 for all k in parallel {
        temp[k] = next[k];
3       while(temp[k]) {
                x[temp[k]] = x[k] + x[temp[k]];
5               temp[k] = temp[temp[k]];
        }
7 }
```

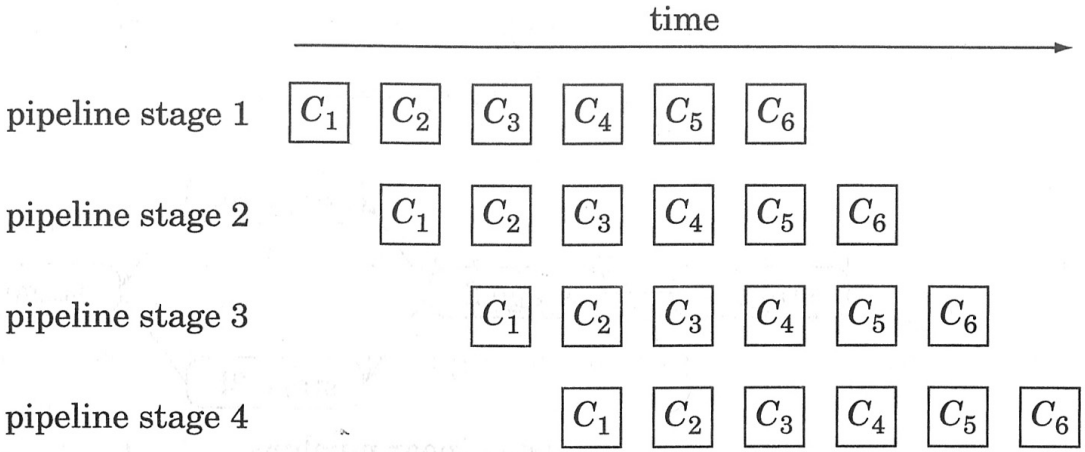
Rekurzivní pohled na data

- Co se děje:



Pipeline

- … známe z procesorů



- Příklady: obecně pravidelné zpracování
  - pipeline instrukcí
  - pipeline na úrovni paralelizace smyček (vektorové stroje)
  - zpracování signálů sérií filtrů
  - rozdělení grafického zpracování na „filtry“
  - vyjádření závislostí na *n* předchozích krocích

Pipeline

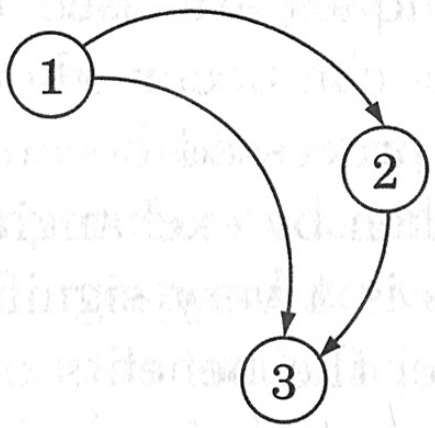
- Efektivita:
  - počet fází pipeline (škálovatelnost)
  - rovnoměrnost výpočetní náročnosti fází
  - plnění a dojíždění pipeline vs. celkový běh
  - průtok dat vs. latence zpracování
    - napojování pipeline, resp. vytvoření co nejdelší pipeline
  - všechny prvky by měly projít pravidelně stejnou sérií zpracování
- Reprezentace toku dat
  - sdílená paměť, soubory – uspořádaná fronta pro každou fázi
  - distribuovaná paměť – bufferované zasílání zpráv
    - v případě vyšší latence přenosu zpráv možno agregovat více požadavků (prodlužuje naplnění pipeline)
  - agregace dat, pokud jich každá fáze potřebuje více (dtto)
- Chyby ve zpracování
  - separátní komponenta sbírající informace chybách

Pipeline

- Plánování úloh
  - možnost/nutnost slučování fází: využívání cache, registrů
  - možnost paralelizace uvnitř fází
  - možnost více instancí jedné fáze
    - pokud nevadí přeuspořádání
- Příklady:
  - zpracování dat ve frekvenční doméně – DFT
    - provedení DFT/FFT
    - aplikace jednotlivých filtrů
    - inverzní DFT/FFT
  - např. zpracování telemetrických dat z Hubblova teleskopu, z radarů AWACS
  - častá implementace v HW – zpracování signálů v reálném čase
  - zpracování HTTP požadavků – filtrování Java Servlety

Událostmi řízená paralelizace

- Nepravidelná a asynchronní obdoba pipeline
- Nepravidelné rozdělené a zpracování vstupů – nevhodné pro SIMD/SIMT



- Problém přeuspořádání událostí
- Obecné komunikační schéma s možností deadlocků