Paralelné programovanie Architektúry

doc. Ing. Michal Čerňanský, PhD.
FIIT STU Bratislava

Prehľad tém

- Implicitný paralelizmus – architektúry porcesorov
- Výkonnostné limitácie pamäťového systému
- Delenie paralelných architektúr
- Komunikačný model paralelných architektúr
- Fyzická organizácia paralelných architektúr
- Komunikačná náročnosť v paralelných systémoch
- Cenové modely zasielania správ a smerovanie
- Techniky mapovania
- Prípadové štúdie

Rozsah a záber paralelizmu

- Konvenčné architektúry – procesor, pamäť, zbernica
- Každý z komponentov - výkonnostné prekážky
- Paralelizmus významným spôsobom prispieva k výkonnosti každého komponentu
- Rôzne aplikácie – rôzne aspekty paralelizmu
  - Dátovo náročné aplikácie – pamäťová priepustnosť
  - Serverové aplikácie – sieťová priepustnosť
  - Vedecké výpočty – vysoké výpočtové a pamäťové nároky
- Potrebné pochopiť každú z týchto výkonnostných prekážok

Implicitný paralelizmus
Architektúry súčasných mikroprocesorov

- Rádové zvýšenie taktovacej frekvencie procesorov
- Významný nárast počtu tranyistorov
- Ako najlepšie využiť zdroje?
- Súčasné procesory – viaceré funkčné jednotky vykonávajúce viaceré inštrukcie súbežne
- Množstvo rôznych prístupov a architektúr

Prúdové spracovanie
Superskalárne procesory

- Prúdové spracovanie (Pipelining) – prekrývanie viacerých štádií spracovania inštrukcie
- Na istej úrovni abstrakcie
  - Kým je jedna inštrukcia je vykonávaná
  - Tak ďalšia inštrukcia je dekódovaná
  - A ďalšia inštrukcia je načítavaná
- Podobnosť s výrobnou linkou v automobilovom priemysle, pásová výroba

Prúdové spracovanie
Superskalárne procesory

- Prúdové spracovanie – významné obmedzenia
- Rýchlosť prúdového spracovania je daná najpomalšou časťou – štádiom spracovania inštrukcie
- Súčasné procesory – hlboká prúdová linka (Pipeline), desiatky štádií
- Na druhej strane – každých 5 až 6 inštrukcií je podmienený skok – potreba kvalitnej predikcie skokov (Branch Prediction)
- Pokuta za zlú predikciu rastie z dĺžkou prúdovej linky (viaceré rozpracované inštrukcie musia byť zahodené)

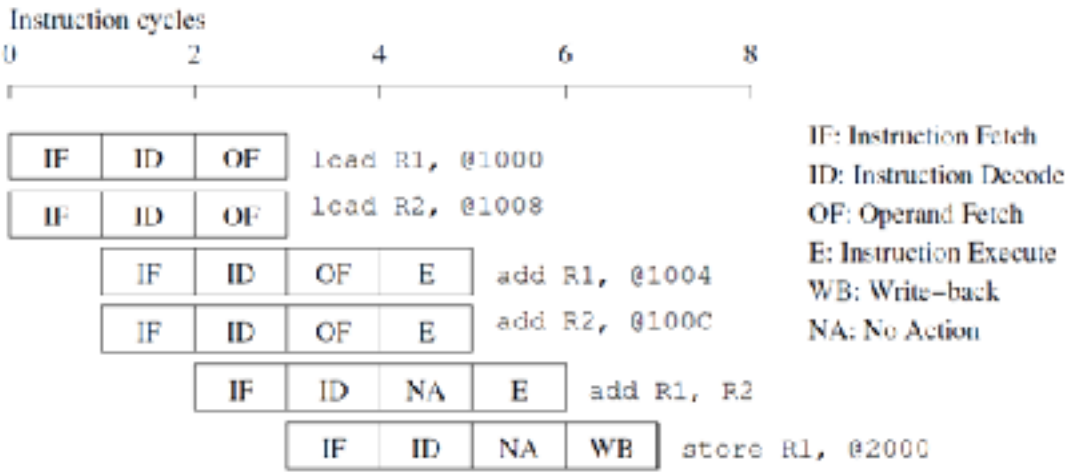
Prúdové spracovanie
Superskalárne procesory

- Jednoduché riešenie, ako zvýšiť priepustnosť – použiť viaceré linky prúdového spracovania
- Ako vyberať inštrukcie?

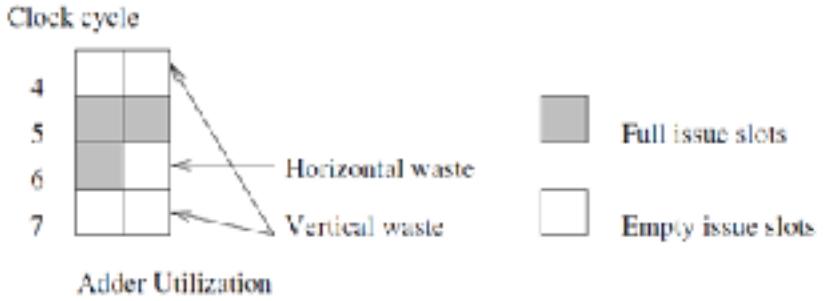
```
1. load R1, @1000        1. load R1, @1000        1. load R1, @1000
2. load R2, @1008        2. add R1, @1004         2. add R1, @1004
3. add R1, @1004         3. add R1, @1008         3. load R2, @1008
4. add R2, @100C         4. add R1, @100C         4. add R2, @100C
5. add R1, R2            5. store R1, @2000       5. add R1, R2
6. store R1, @2000                                6. store R1, @2000
```

(i) (ii) (iii)

(a) Three different code fragments for adding a list of four numbers.



(b) Execution schedule for code fragment (i) above.



(c) Hardware utilization trace for schedule in (b).

Prúdové spracovanie
Superskalárne procesory

- Plytvanie zdrojmi – dátová závislosť medzi inštrukciami
- Rôzne sady inštrukcií s rovnakou sémantikou – rôzny čas ich vykonania

Prúdové spracovanie
Superskalárne procesory

- Plánovanie inštrukcií závisí od:
- Skuťočná dátová závislosť – výsledok jednej operácie je vstupom do druhej operácie
- Závislosť na zdroji – dve operácie závisia rovnakom zdroji (napr. FPU)
- Závislosť na výsledkoch podmienok v inštrukciách skoku
- Plánovač – súčasť HW procesora, zisťuje, ktoré inštrukcie z inštrukčného toku môžu byť vykonané súbežne na základe uvedených faktorov
- Zložitosť tohto HW je významná (desiatky percent! z plochy procesora)

Prúdové spracovanie
Superskalárne procesory

- Jednoduchý model
  - Vykonávanie inštrukcií v poradí danom tokom inštrukcií
  - Ak je druhá inštrukcia dátovo závislá na prvej, iba jedna inštrukcia v cykle (in-order vykonávanie)
  - Obmedzená výkonnosť
- Agresívnejšie výkonávanie
  - Vykonávanie inštrukcií aj mimo poradia (out of order)
  - Ak je druhá inštrukcia dátovo závislá na prvej a tretia inštrukcia nie je závislá, je možné naplánovať vykonanie prvej a tretej inštrukcie súbežne

Prúdové spracovanie
Superskalárne procesory

- Všetky funkčné jednotky nie sú vyťažené v každom cykle
- Žiadna nie je využitá v cykle – vertikálne plytvanie (vertical waste)
- Ak iba niektoré nie sú využité v cykle – horizontálne plytvanie (horizontal waste)
- Limitácia superskalárnych procesorov
  - Limitovaný paralelizmus v inštrukciách (závislosti)
  - Neschopnosť plánovača extrahovať paralelizmus
- Bežné procesory – štvorcestné prúdové spracovanie

Prúdové spracovanie
Superskalárne procesory

- Superskálárne procesory – drahý a náročný hardvér spojený s plánovaním inštrukcií
- Procesory s veľmi dlhým inštrukčným slovom (Very Large Instruction World, VLIW)
- Analýza počas kompilácie
- Zakódovanie viacerých operácií (‚ktoré môžu byť vykonané súbežne) do jedného inštrukčného slova

Prúdové spracovanie
Superskalárne procesory

- Jednoduchší a lacnejší HW
- Kompilátor ma bohatší kontext na rozhodnutie, ktoré inštrukcie naplánovať na súbežné vykonanie
- Kompilátor nemá informácie o behu vykonávania, napr. výpadok vo vyrovnávacej pamäti – konzervatívne plánovanie
- Náročnejšia predikcia skokov a pamäti
- Závislosť VLIW architektúr od kompilátora, potreba kvalitného rozbaľovania cyklov (loop unrolling), špekulatívneho vykonávania inštrukcií predickie skokov
- Typická VLIW architektúra procesora – 4 až 8 cestný paralelizmus

Obmedzenia pamäťového systému

- Pamäťový systém je často úzkym hrdlom z hľadiska výkonu pre mnohé aplikácie (nie rýchlosť procesora)
- Výkonnostné vlastnosti pamäťového systému – doba odozvy (prístupová doba, oneskorenie) a priepustnosť (latency and bandwidth)
- Prístupová doba – čas od požiadavky na dáta až kým sú dáta prístupné
- Priepustnosť - množstvo dát prenesených z pamäte do procesora za jednotku času

Obmedzenia pamäťového systému

- Rozdiel medzi prístupovou dobou a priepustnosťou
- Hydrant
- Doba odozvy - čas od otočenia kohútika až kým nezačne tiecť voda (napr. 2s)
- Priepustnosť - množstvo vody vytečenej za jednotku času (10l za s) It is very important to understand the difference between latency and bandwidth.
- Potreba okamžitej reakcie – znížiť dobu odozvy
- Potreba hasiť veľké požiare – zvýšiť priepustnosť

Obmedzenia pamäťového systému

- Procesor operujúci s taktovacou frekvenciou 1GHz (1ns cyklus = takt)
- Pripojený na RAM s latenciou 100ns (žiadna cache)
- Procesor obsahuje 2 FPU jednotky (vykonávajúce inštrukcie „vynásob a pričítaj“ - multiply-add) a je schopný vykonať 4 inštrukcie v každom 1ns cykle
  - 4 GFLOPS – max. FP výkon procesora
  - Latencia pamäte – 100 cyklov, údaje čítané po blokoch o veľkosti jedného slova, procesor musí čakať 100 cyklov kým môže spracovať dáta

Obmedzenia pamäťového systému

- Výpočet skalárneho súčinu (dot product) dvoch vektorov na uvedenej architektúre
- Jedna inštrukcia multiply-add na jednom páre položiek vektorov, každá hodnota vyžaduje jedno vyčítanie z pamäte
- Jedna FP inštrukcia každých 100 ns, čiže výsledných 10 MFLOPS, a to je iba zlomok teoretického výkonu

Obmedzenia pamäťového systému

- Vyrovnávacia pamäť – malá a rýchla pamäť medzi procesorom a hlavnou pamäťou
- Úložisko s malou latenciou a vysokou priepustnosťou
- Znižuje efektívnu latenciu pamäťového systému, ak sú údaje vo vyrovnávacej pamäti opakovane používané
- „Cache hit ratio“ – pomer prístupov, pri ktorých sa údaje nachádzajú vo vyrovnávacej pamäti a nie je potrebný prístup do hlavnej pamäti (ku všetkým prístupom)

Obmedzenia pamäťového systému

- Opakovaný prístup k tým istým údajom – časová lokálnosť (temporal locality)
- Násobenie matíc $O(n^2)$ prístupov do hlavnej pamäte ale $O(n^3)$ výpočtov
- Opakované použitie údajov – kritické pre výkonnosť vyrovnávacej pamäte

Obmedzenia pamäťového systému

- Architektúra z predchádzajúceho príkladu
- Vyrovnávacia pamäť s veľkosťou 32 KB s tatenciou 1 ns (1 cyklus)
- Násobenie matíc C = A x B s rozmermi 32 x 32 (vyrovnávacia pamäť dostatočne veľká na uloženie matíc A,B a C)

Obmedzenia pamäťového systému

- Načítanie 2 matíc do vyrovnávacej pamäti zodpovedá načítaniu 2k slov (2x32x32), čo trvá 200 μs
- Násobenie dvoch n x n matíc si vyžaduje $2n^3$ FP operácií, v našom prípade 64K operácií vykonateľných za 16k cyklov (4 inštrukcie za cyklus, spolu 16 μs)
- Celkový čas výpočtu je 200 + 16 μs
- To zodpovedá výkonu 64k operácií za 216 μs - 64K/216 = 303 MFLOPS

Obmedzenia pamäťového systému

- Opakovaný prístup k tým istým údajom – časová lokálnosť (temporal locality)
- Násobenie matíc $O(n^2)$ prístupov do hlavnej pamäte ale $O(n^3)$ výpočtov
- Opakované použitie údajov – kritické pre výkonnosť vyrovnávacej pamäte

Obmedzenia pamäťového systému

- Priepustnosť pamäťového systému je daná pamäťovou zbernicou aj pamäťovými jednotkami
- Priepustnosť môže byť zlepšená zvýšením veľkosti pamäťových blokov
- i časových jednotiek potrebných na prenesenie b dátových jednotiek (i – latencia, b – veľkosť prenášaného bloku)

Obmedzenia pamäťového systému

- Predchádzajúci prípad so skalárnym súčinom ale veľkosť bloku 4 slová namiesto 1 slova
- Vektory lineárne v pamäti – 8 FLOPs (4x multiply-add) za 200 cyklov
- Jeden prístup do pamäte - 4 položky vektora
- Dva prístupy do pamäte – 4 položky každého vektora, čo zodpovedá 1 FLOP každých 25 ns, čiže výkon 40 MFLOPS

Obmedzenia pamäťového systému

- Zvyšovanie veľkosti prenášaných blokov nemení latenciu pamäťového systému
- Fyzicky je možné si situáciu predstaviť ako široká pamäťová zbernica o 4 slovách (128 bitov) pripojená na viaceré pamäťové banky
- Finančne nákladná architektúra
- Praktickejšie realizácie – viaceré slová zaslané v nasledujúcich cykloch

Obmedzenia pamäťového systému

- Uvedený príklad demonštruje ako zvýšená priepustnosť môže zlepšiť výpočtový výkon počítačového systému
- Susedné položky v pamäti použité „susednými“ inštrukciami – priestorová lokálnosť (spatial locality)
- Z hľadiska uloženia údajov v pamäti – potreba preskupiť operácie výpočtu tak, aby sa čo najviac využila priestorová lokálnosť

Obmedzenia pamäťového systému

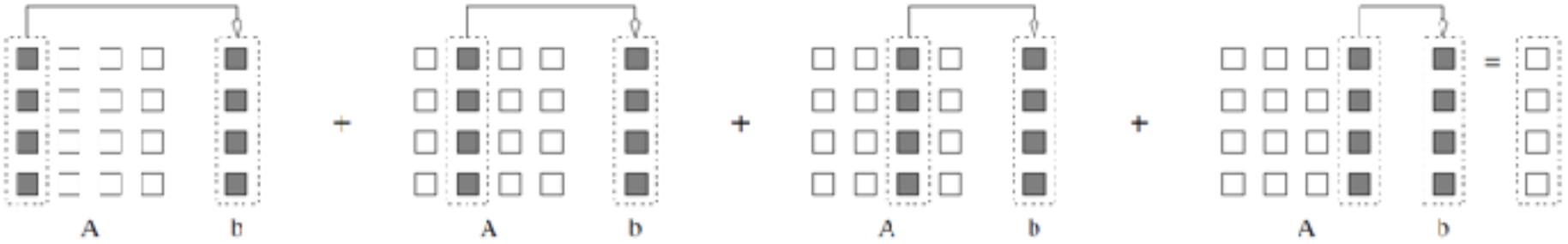
- Uvažujme nasledujúci snippet (snáď znovupoužiteľný kúsok kódu)

```
for (i = 0; i < 1000; i++)
    column_sum[i] = 0.0;
        for (j = 0; j < 1000; j++)
         column_sum[i] += b[j][i];
```

- Tento fragment kódu vypočítava súčty stĺpcov matice `b` do vektora `column_sum`

Obmedzenia pamäťového systému

- Vektor `column_sum` sa zmestí do vyrovnávacej pamäte
- K matici `b` je pristupované po stĺpcoch
- „Prekladaný“ prístup spôsobuje nepriaznivú výpočtovú výkonnosť



(a) Column major data access

(b) Row major data access.

Obmedzenia pamäťového systému

- Úprava pôvodného kódu:

```
for (i = 0; i < 1000; i++)
          column_sum[i] = 0.0;
for (j = 0; j < 1000; j++)
          for (i = 0; i < 1000; i++)
           column_sum[i] += b[j][i];
```

- Prvky matice sú prechádzané po riadkoch a výsledná výpočtová výkonnosť môže byť výrazne lepšia

Obmedzenia pamäťového systému

- Príklady demonštrujú nasledujúce koncepty:
  - Využívanie priestorovej a časovej lokálnosti je kritické pre zvyšovanie efektívnej pamäťovej priepustnosti a zakrývaní prístupovej doby
  - Pomer počtu výpočtových operácií k počtu prístupov do pamäte je dobrým indikátorom náchylnosti na obmedzenie vyplývajúce z pamäťovej priepustnosti
  - Rozloženie dát v pamäti a zodpovedajúca organizácia výpočtov môže významným spôsobom ovplyvniť priestorovú a časovú lokánlosť

Obmedzenia pamäťového systému

- Alternatívne prístupy znižujúce obmedzenia pamäťového systému
- Browsovanie webu na pomalom dátvom pripojení
  - Predvídať, ktoré stránky nás zaujímajú a dopredu požadovať ich stiahnutie (Prefetching)
  - Otvoriť viaceré prehliadače a v každom z nich pristupovať k inej stránke, zatiaľ čo na jednom si stránku pozeráme, ostatné môžu sťahovať stránky (Multithreading)
  - Požadovať stiahnutie viacerých stránok naraz, amortizujúc čas prístupu medzi viaceré prístupy (Spatial locality)

Obmedzenia pamäťového systému

- Viacvláknovosť (Mulithreading)
- Vlákno – jeden prúd riadenia (a vykonávania) v rámci programu
- Jednoduchý príklad:

```
for (i = 0; i < n; i++)
    c[i] = dot_product(get_row(a, i), b);
```

- Nezávislosť funkcií `dot_product` , ktoré môžu reprezentovať súbežne vykonateľné jednotky:

```
for (i = 0; i < n; i++)
    c[i] = create_thread(dot_product,get_row(a, i), b);
```

Obmedzenia pamäťového systému

- V predchádzajúcom príklade prvá inštancia funkcie pristupovala k páru zodpovedajúcich položiek vektorov a čaká na ne
- Počas toho druhá inštancia tejto funkcie pristupuje k ďalším prvkom vektorov v ďalšom cykle, atď...
- Po i časových jednotkách, kde i je latencia pamäťového systému, prvá inštancia funkcie získa požadované údaje z pamäte a môže vykonať požadovaný výpočet
- V ďalšom cykle sú k dispozícii už údaje pre ďalšiu inštanciu funkcie, atď..
- Takýmto spôsobom je v každom cykle vykonaný výpočet

Obmedzenia pamäťového systému

- Vykonanie operácií naplánovaných podľa predchádzajúceho príkladu závisí od splnenia dvoch predpokladov: pamäťový systém je schopný obslúžiť viaceré súbežné požiadavky a procesor je schopný prepnúť vykonávanie medzi vláknami v každom cykle
- Explicitné určenie paralelizmu vo forme vlákien
- Moderné GPU

Obmedzenia pamäťového systému

- Výpadky vo vyrovnávacej pamäti (cache misses) – zastavenie programu
- Skoré načítanie údajov tak, aby v čase ich potreby už boli k dispozícií
- Potreba ďalšieho priestoru
- Prepísanie skôr „predčítaných“ údajov – nie horšie ako bez predčítavania (prefetching-u)

Obmedzenia pamäťového systému

- Viacvláknovosť (multithreading) a predčítavanie (prefetching) sú silne obmedzené pamäťovou priepustnosťou
- Výpočtový systém
  - Frekvencia 1GHz, 4 cestná vyrovnávacia pamäť, prístup do vyrovnávacej pamäte 1 cyklus, 100ns latencia prístupu do RAM, Cache Hit Ratio pri 1KB je 25% a pri 32 KB of 90%
- Dva prípady
  - 1 vlákno má k dispozícií celú vyrovnávaciu pamäť
  - 32 vlákien každé s 1KB
- Nároky na pamäťovú priepustnosť keď požiadavka na pamäť v každom cykle
  - 1 vlákno - 400MB/s
  - 32 vlákien - 3GB/s.

Obmedzenia pamäťového systému

- Požiadavky na vysokú priepustnosť môžu významne narásť pri viacvláknových systémoch (nižšia „cache residency“)
- Viacvláknové systémy sa stávajú limitovanými priepustnosťou a nie latenciou
- Viacvláknovosť a skoré načítavanie (prefetching) sa snažia riešiť iba problém s latenciou a môžu prehĺbiť problém s priepustnosťou
- Viacvláknovosť a skoré načítavanie tiež vyžadujú podstatne viac HW zdrojov vo forme pamäťového priestoru

Dichotómia paralelných platforiem

- Explicitný paralelizmus – program musí definovať súbežnosť a interakciu medzi paralelnými úlohami
- Štruktúra riadenia (Control Structure) – definovanie súbežnosti
- Model komunikácie (Communication Model) – definovanie interakcie medzi úlohami

Dichotómia paralelných platforiem

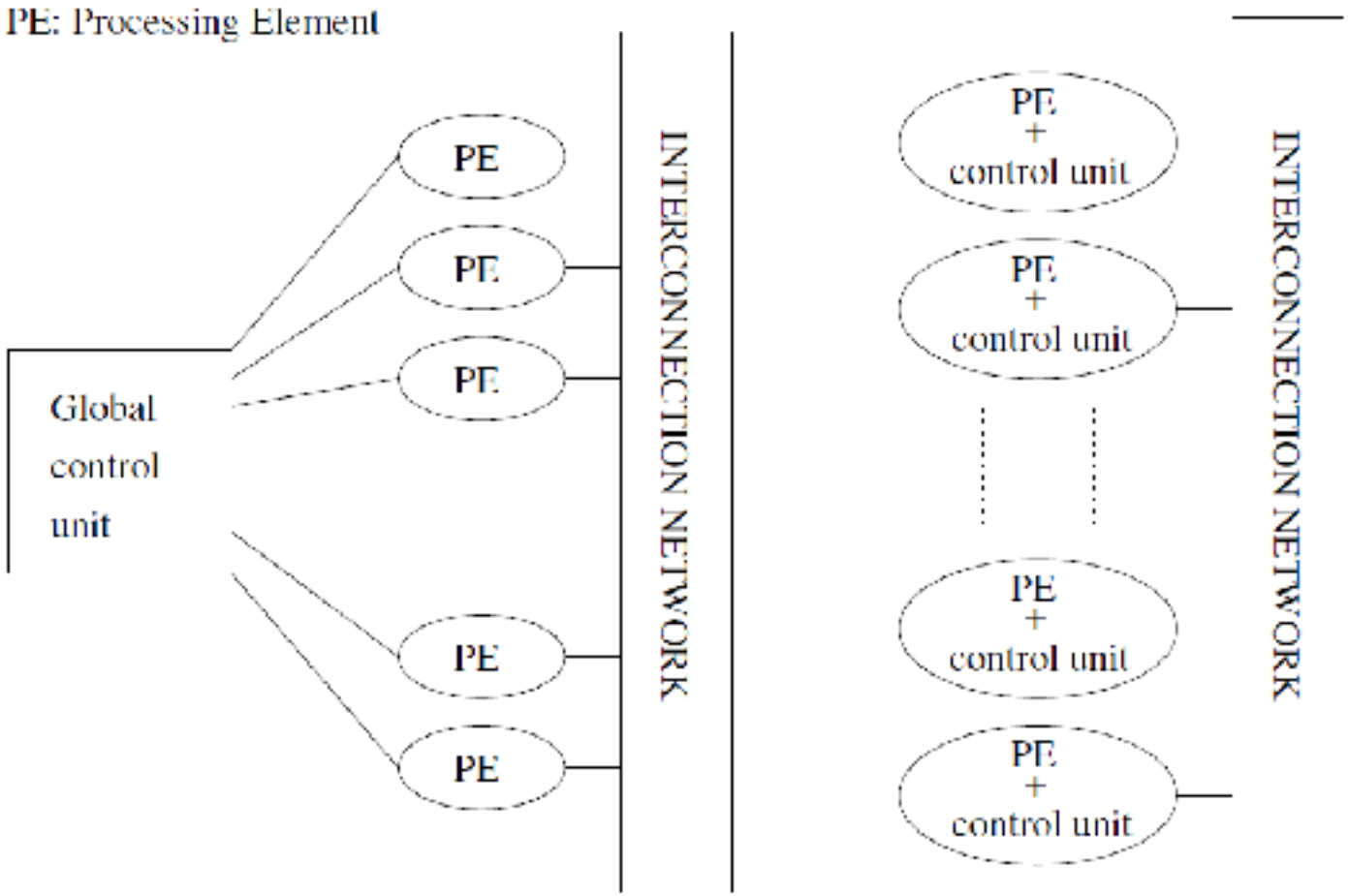
- Štruktúra riadenia (Control Structure)
  - Paralelizmus sa môže prejaviť na rôznych úrovniach – od inštrukcií až na úroveň procesov
  - Mnohé modely medzi týmito extrémami, spolu s architektonickou podporou

Dichotómia paralelných platforiem

- Štruktúra riadenia (Control Structure)
  - Výkonné jednotky v paralelných počítačových systémoch sú riadené jednou centralizovanou riadiacou jednotkou alebo má každá vlastnú riadiacu jednotku
  - Ak jediná centrálna riadiaca jednotka tou istou inštrukciou riadi viaceré výkonné jednotky – SIMD model (single instruction stream, multiple data stream)
  - Ak má každá výkonná jednotka vlastnú riadiacu jednotku, každý takýto procesor môže vykonať rôzne inštrukcie na rôznych dátach – MIMD (multiple instruction stream, multiple data stream)

Dichotómia paralelných platforiem

- SIMD a MIMD systémy

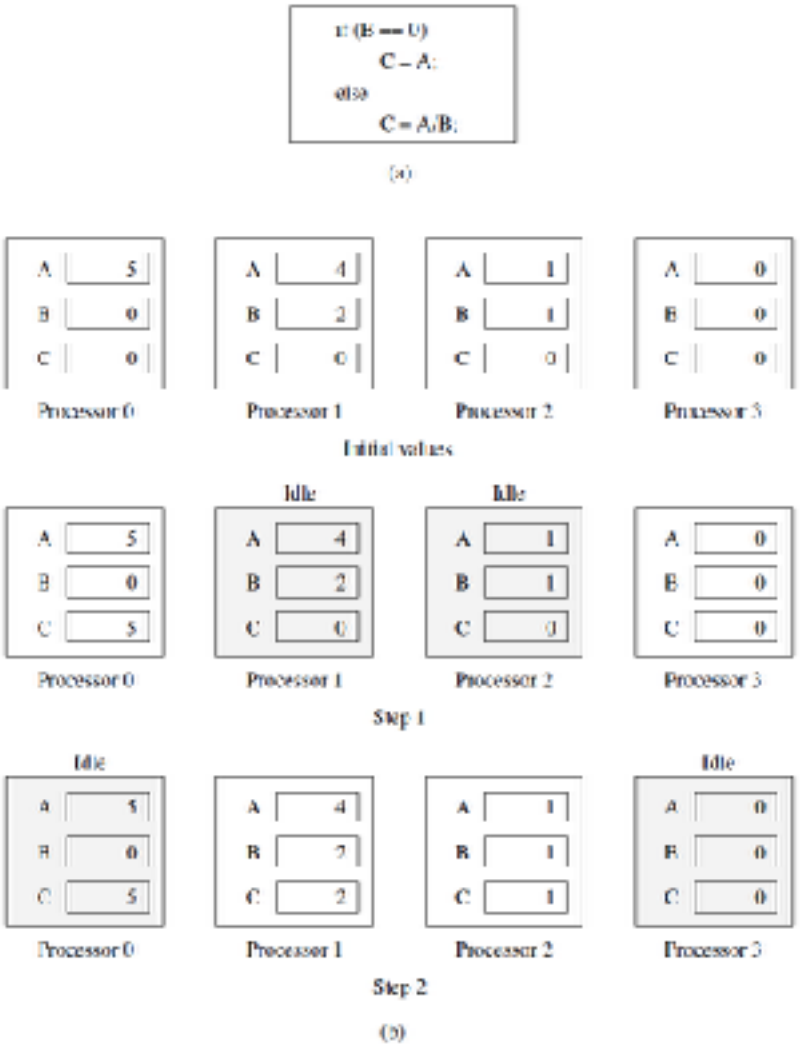


Dichotómia paralelných platforiem

- SIMD systémy
  - Staršie paralelné počítače(Illiac IV, MPP, DAP, CM-2, and MasPar MP-1)
  - Variant konceptu sa uplatnil v tzv. vektorových inštrukciach (sady inštrikcií, MMX)
  - SIMD systémy sa spoliehajú na pravidelnú štruktúru vo výpočtoch (napr. spracovanie obrazu)
  - Možnosť selektívne vybrať dátové položky, nad ktorými sa má operácia vykonať – activity mask

Dichotómia paralelných platforiem

- Podmienené vykonanie v SIMD systémoch



Dichotómia paralelných platforiem

- MIMD systémy
  - Na rozdiel od SIMD systémov MIMD systémy môžu vykonávať rôzne programy na rôznych procesoroch
  - SPMD (single program multiple data streams) - variant SIMD, vykonanie rovnakého programu na rôznych procesoroch
  - NOW (Networks of Workstations), počítačové klastre, viacjadrové počítačové systémy, grafické procesory

Dichotómia paralelných platforiem

- Porovnanie SIMD a MIMD
- SIMD – menej náročný HW (stačí jediná riadiaca jednotka)
- SIMD – špecializované určenie, náročnejší vývoj, dlhý vývojový cyklus
- Menej aplikácií je vhodných pre SIMD platformy
- MIMD – možnosť vybudovať z klasických SISD (alebo SIMD) komponentov - finančne priaznivé

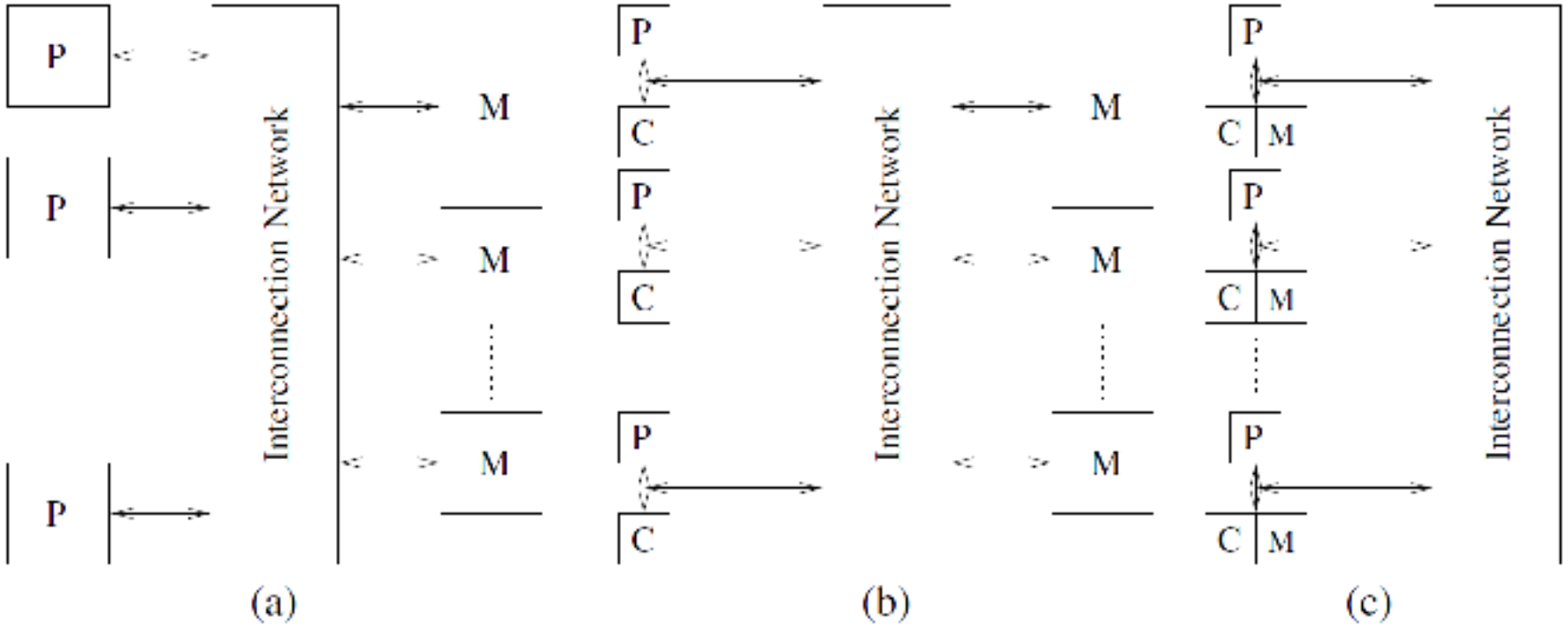
Dichotómia paralelných platforiem

- Dva základné prístupy k výmene údajov medzi paralelnymi úlohami
  - Prístup do spoločnej pamäte
  - Výmena správ
- Platformy umožňujúce prístup do spoločného dátového priestoru – systémy so zdieľaným pamäťovým priestorom (shared-address-space) - multiprocesory
- Platformy podporujúce výmenu správ – platformy zasielania správ (message passing) - multipočítače

Dichotómia paralelných platforiem

- Systémy so spoločným pamäťovým priestorom
- Časť alebo celá pamäť je prístupná všetkým procesorom
- Procesory interagujú modifikovaním údajov v spoločnom pamäťovom priestore
- Ak je čas prístupu na každé miesto v pamäti rovnaký – UMA (uniform memory access), inak NUMA (non-uniform memory access)

Dichotómia paralelných platforiem



Dichotómia paralelných platforiem

- Rozdiel medzi NUMA a UMA platformami je dôležitý vzhľadom na návrh algoritmov
- NUMA systémy vyžadujú využiť lokálnosť problému a algoritmu na dosiahnutie výkonnosti
- Jednoduchšie programovanie systémov so zdieľaným pamäťovým priestorom, čítanie a zapisovanie do pamäte implicitne viditeľné z každého procesora
- Potreba koordinovať čítanie a zápisy
- Použitie vyrovnávacích pamätí vyžaduje koordináciu prístupu k viacerým kópiám údajov – problém s koherenciou vyrovnávacej pamäte
- Slabší model poskytuje pamäťový priestor, ale prístup do vyrovnávacích pamätí nie je koordinovaný (non cache coherent shared address space machines)

Dichotómia paralelných platforiem

- Systémy so zdieľaným pamäťovým priestorom vs. systémy so zdieľanou pamäťou
  - Rozdiel v pojmoch
  - Systémy so zdieľaným pamäťovým priestorom – abstrakcia
  - Systémy so zdieľanou pamäťou – fyzická organizácia pamäťovej architektúry systému
  - Je možne poskytovať zdieľaný pamäťový priestor na systémoch s fyzicky distribuovanou pamäťou

Dichotómia paralelných platforiem

- Platformy zasielania správ
- Platformy sú zložené z procesorov obsahujúcich vlastnú pamäť
- Počítačové klastre, multipočítače bez zdieľaného adresového priestoru
- Platformy využívajú varianty komunikačých primitív send a receive
- Napr. knižnice MPI a PVM poskytujú tieto primitívy

Dichotómia paralelných platforiem

- Zasielanie správ vs. zdieľaný priestor adries
- Zasielanie správ si vyžaduje malú HW podporu, stačí sieťový subsystém
- Platformy so zdieľaným priestorom adries môžu jednoducho emulovať zasielanie správ
- Opačný prístup (emulácia spoločného pamäťového priestoru na systémoch zasielania sprav) je podstatne náročnejšia

Fyzická organizácia paralelných systémov

- RAM - Random Access Machine, výpočtový model klasického počítača
- PRAM – Parallel RAM
  - P procesorov – rovnako taktované
  - Hlavná spoločná pamäť - neohraničéná veľkosť

- EREV PRAM - Exclusive-read, exclusive-write
- CREV PRAM - Concurrent-read, exclusive-write
- ERCW PRAM - Exclusive-read, concurrent-write
- CRCW PRAM - Concurrent-read, concurrent-write

Fyzická organizácia paralelných systémov

- Súbežné čítanie – nie je potrebné špeciálne riešiť
- Súbežné zapisovanie – potreba riadiť prístup
  - Všetky procesory, či sa snažia zapísať na dané miesto musia zapisovať rovnakú hodnotu
  - Iba jednému procesoru sa podarí zápis, ostatné zápisy zlyhajú
  - Podľa priority pridelenej procesoru
  - Vykoná sa redukčná operácia (súčet, iná asociatívna operácia)

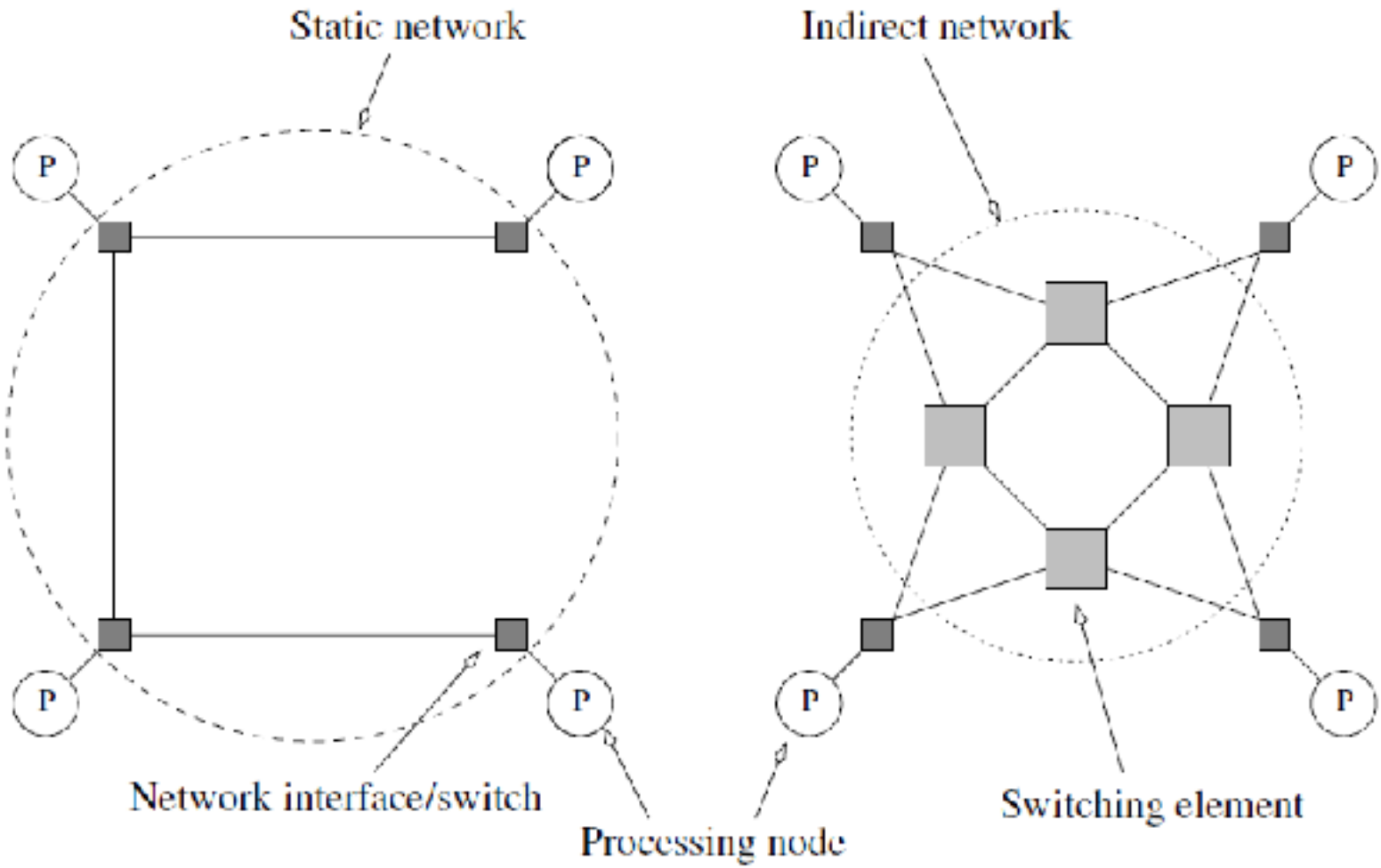
Fyzická organizácia paralelných systémov

- **Náročnosť realizácie ideálnej PRAM architektúry**
- EREV PRAM – p procesorov a m pamäťových miest
  - Procesor pripojený k pamäti cez systém prepínačov
  - Prepínače určujú, ktoré z pamäťových miest sú pripojené na jednotlivé procesory
  - Každý z procesorov môže pristúpiť na ľubovoľné pamäťové miesto, ak už k nemu nepristupuje iný procesor
  - Počet prepínačov úmerný m x p
  - Pre rozumne veľkú pamäť veľmi zložitá a drahá sieť preinačov

Fyzická organizácia paralelných systémov

- Prepojovacie siete – prenos dát medzi procesormi a pamäťovými modulmi
- Model prepojovacej siete – n vstupov a m výstupov
- Výstupy môžu byť totožné so vstupmi
- Prepojovacie siete – prepojenia a prepínače
- Prepojovacie siete
- Statické siete – komunikačný prepojenie typu bod – bod, priame siete
- Dynamické siete – zložené aj z prepínačov umožňujúcich vytváranie komunikačných ciest medzi uzlami a pamäťovými bankami, nepriame siete

Fyzická organizácia paralelných systémov



Fyzická organizácia paralelných systémov

- Prepínač
  - Mapovanie zo vstupných na výstupné porty
  - Vnútorný buffer
  - Smerovanie – zabránenie zahlteniu siete
  - Muticast – zasielanie tých istých údajov na viaceré porty
  - Mapovanie realizované rôznymi mechanizmami
- Sieťové rozhranie
  - Zabezpečuje prepojenie medzi uzlom a sieťou
  - Zabalenie údajov do paketov, informácia pre smerovanie dát, buffrovanie vstupov a výstupov vzhľadom na rýchlosti siete a spracovateľského uzla, kontrola chýb

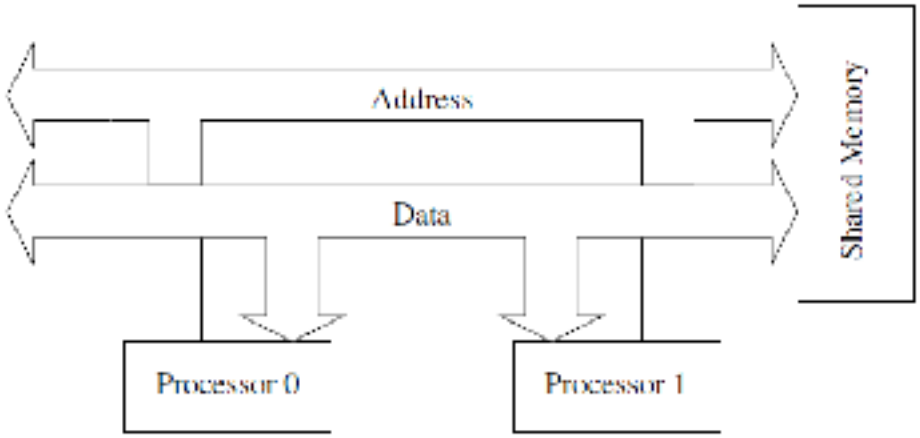
Fyzická organizácia paralelných systémov

- Sieťové topológie - škálovateľnosť, výkonnosť, cena
  - Zbernica
  - Krížový prepínač
  - Viacúrovňové prepojovacie siete
  - Plne-prepojené siete
  - Hviezdicová sieť
  - 2D a 3D Mesh (napr. tórus), hyperkocka
  - Stromové topológie

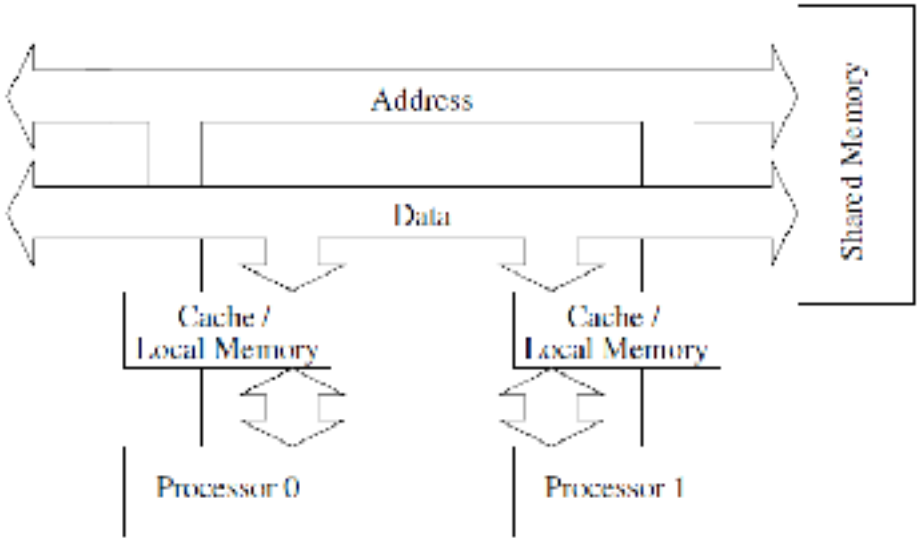
- Komerčné prístupy – hybridné topológie

Fyzická organizácia paralelných systémov

- **Zbernicová topológia**
- Všetky procesory pristupujú na spoločnú dátovú zbernicu
- Vzdialenosť medzi každými dvoma uzlami je O(1)
- Jednoducho realizovateľný broadcast
- Najväčším problémom je priepustnosť
- Typicky desiatky uzlov


Address
Data
Shared Memory
Processor 0
Processor 1

(a)


Address
Data
Shared Memory
Cache /
Local Memory
Cache /
Local Memory
Processor 0
Processor 1

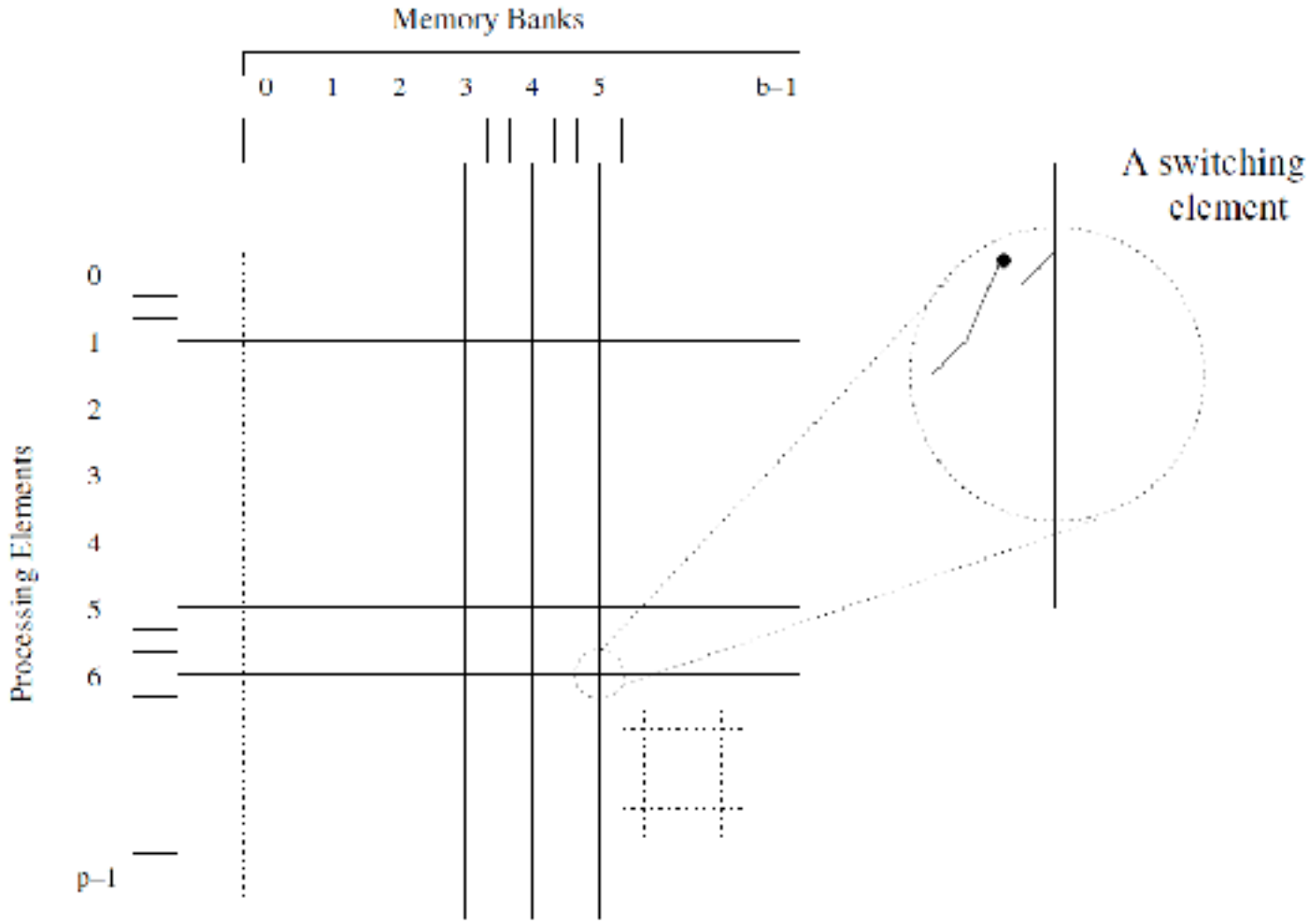
(b)

Fyzická organizácia paralelných systémov

- **Krížový prepínač**
- Mriežka p x b
- p vstupov (procesorov)
- b výstupov (pamäťových modulov)
- Neblokujúce prepojenie

- Zložitosť rastie s $O(p\ b) = O(b^2)$

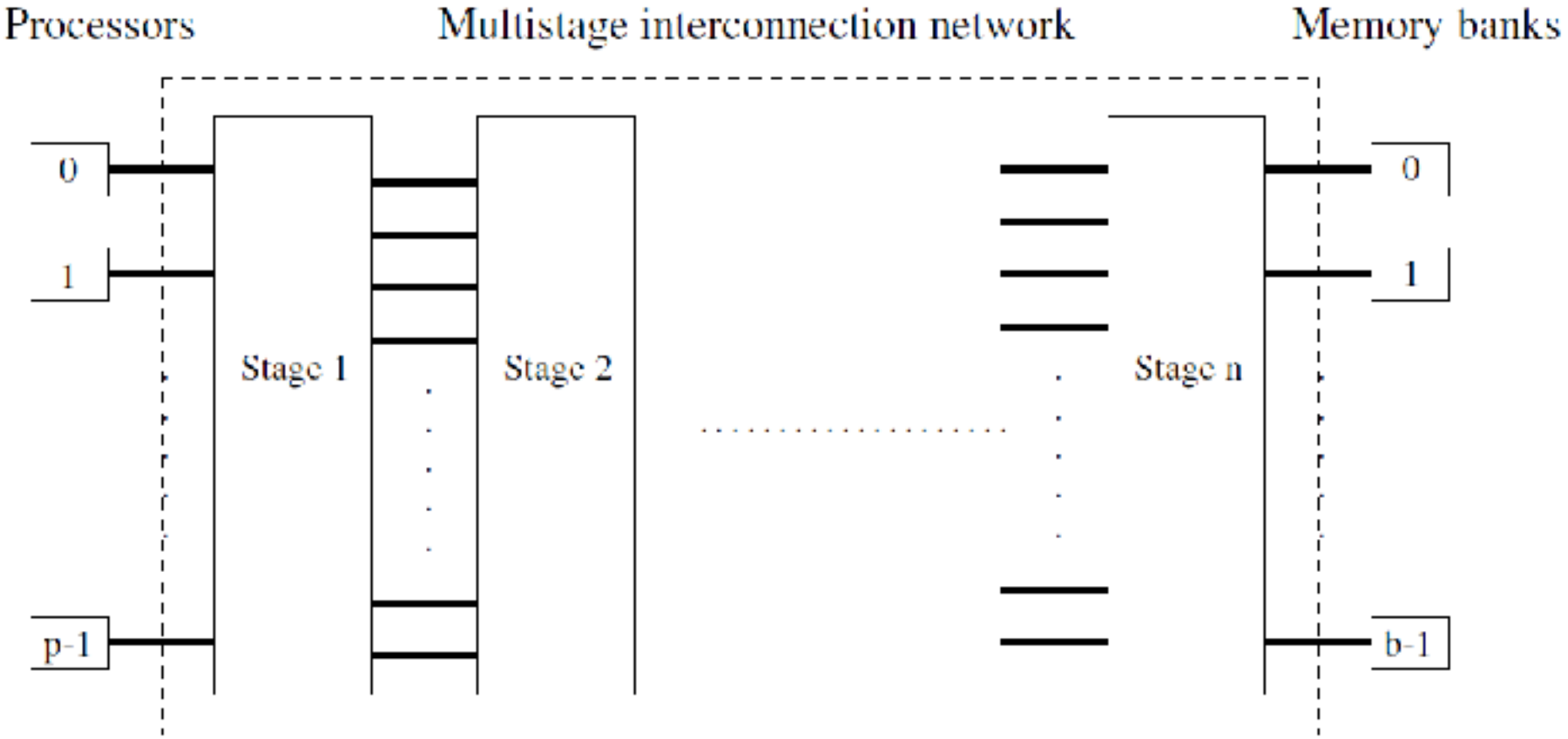
Fyzická organizácia paralelných systémov



Fyzická organizácia paralelných systémov

- **Viacúrovňové prepojovacie siete**
- Kombinácia prístupov
  - Zbernica – lacná ale málo výkonná
  - Krížový prepínač – drahý ale výkonný

Fyzická organizácia paralelných systémov

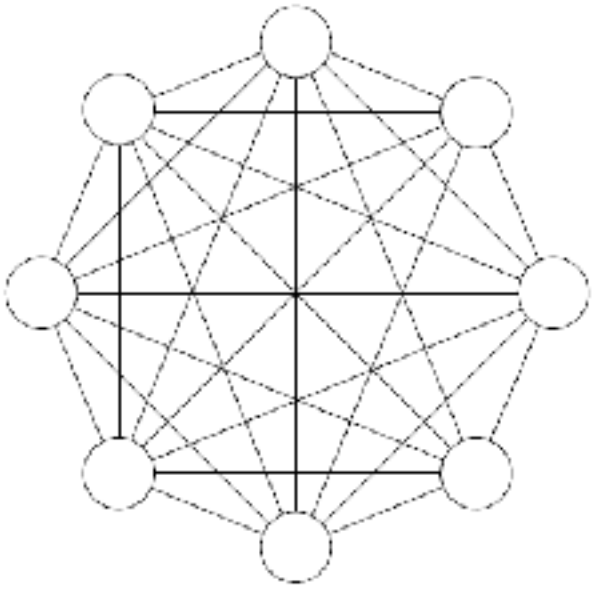


Fyzická organizácia paralelných systémov

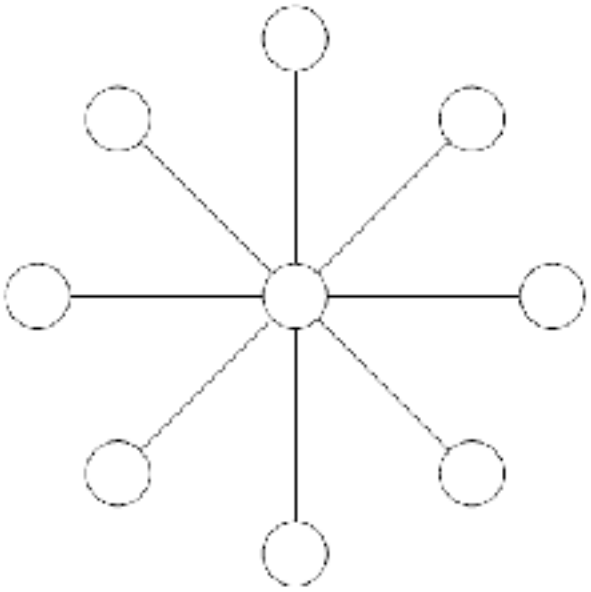
- **Plne prepojená sieť**
- Všetky procesory prepojené navzájom
- Vysoká HW náročnosť v prípade veľkého počtu procesorov
- Statický ekvivalent krížového prepínača

- **Hviezdicová topológia**
- Každý´uzol prepojený na centrálny uzol
- Vzdialenosť medzi všetkými uzlami je *O(1)*
- Centrálny uzol je úzke hrdlo komunikácie
- Statický ekvivalent zbernice

Fyzická organizácia paralelných systémov



(a)



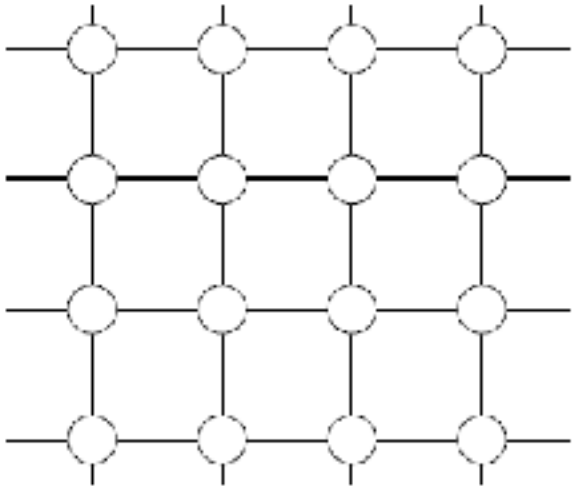
(b)

Fyzická organizácia paralelných systémov

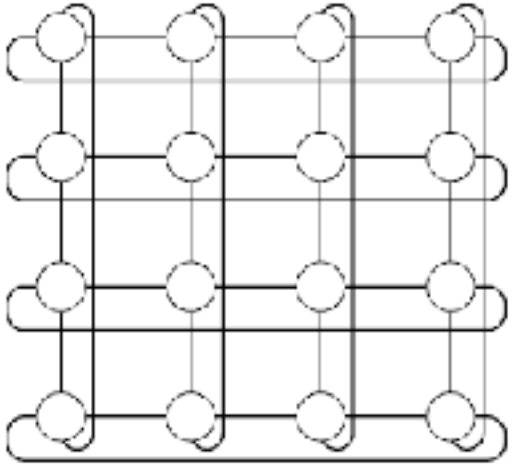
- **2D a 3D mesh topológia**
- Uzly prepojené so svojimi susedmi v mriežke (d je dimenzia, 2d susedov)

- **Hyperkocka**
- Špecialny prípad d-dimenyionalnej mesh topológie

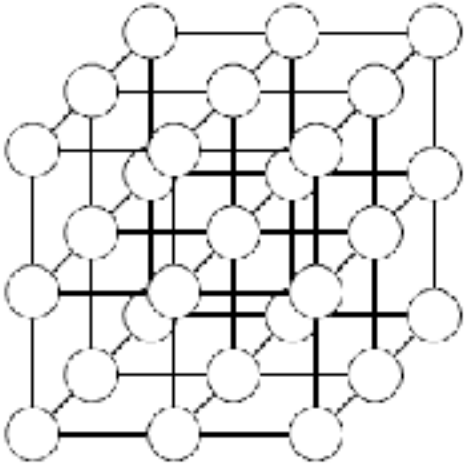
Fyzická organizácia paralelných systémov



(a)

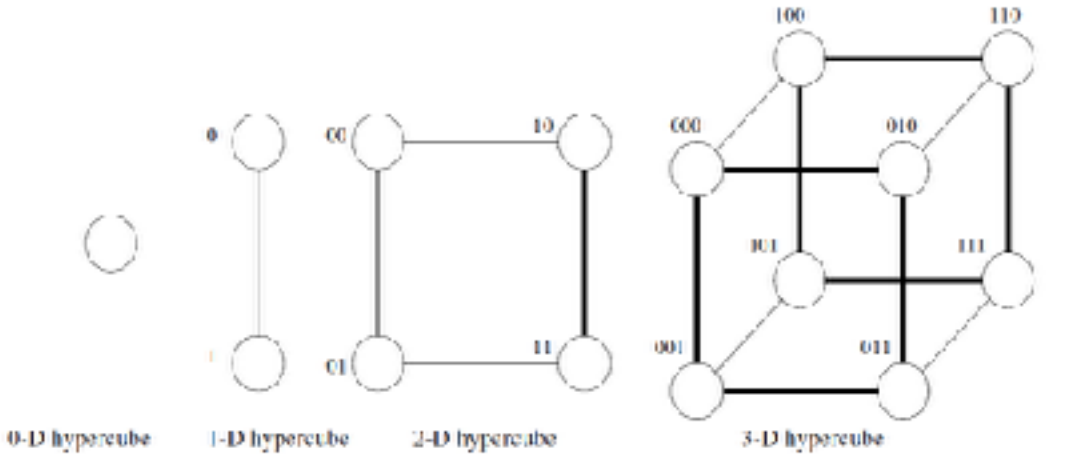


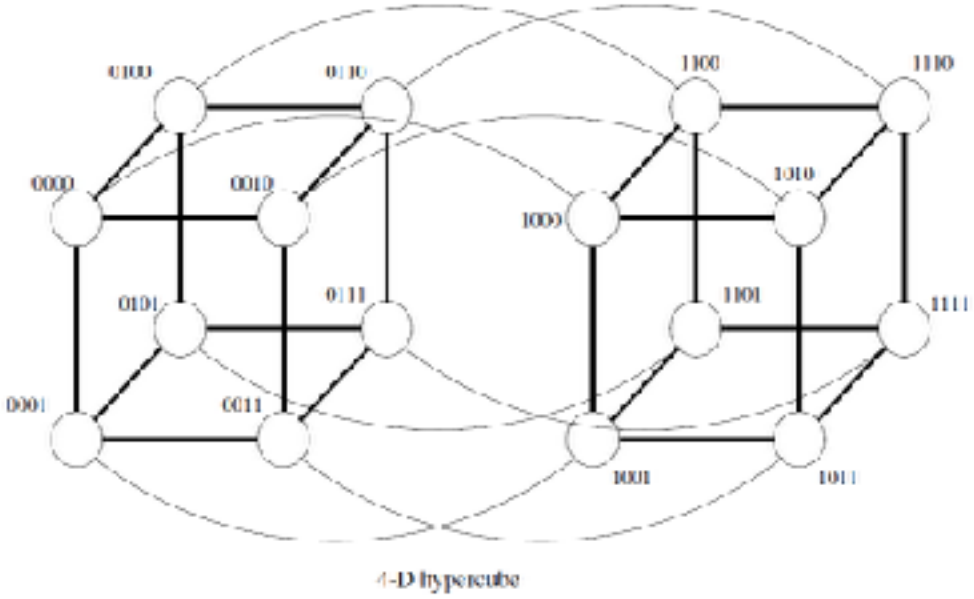
(b)



(c)

Fyzická organizácia paralelných systémov

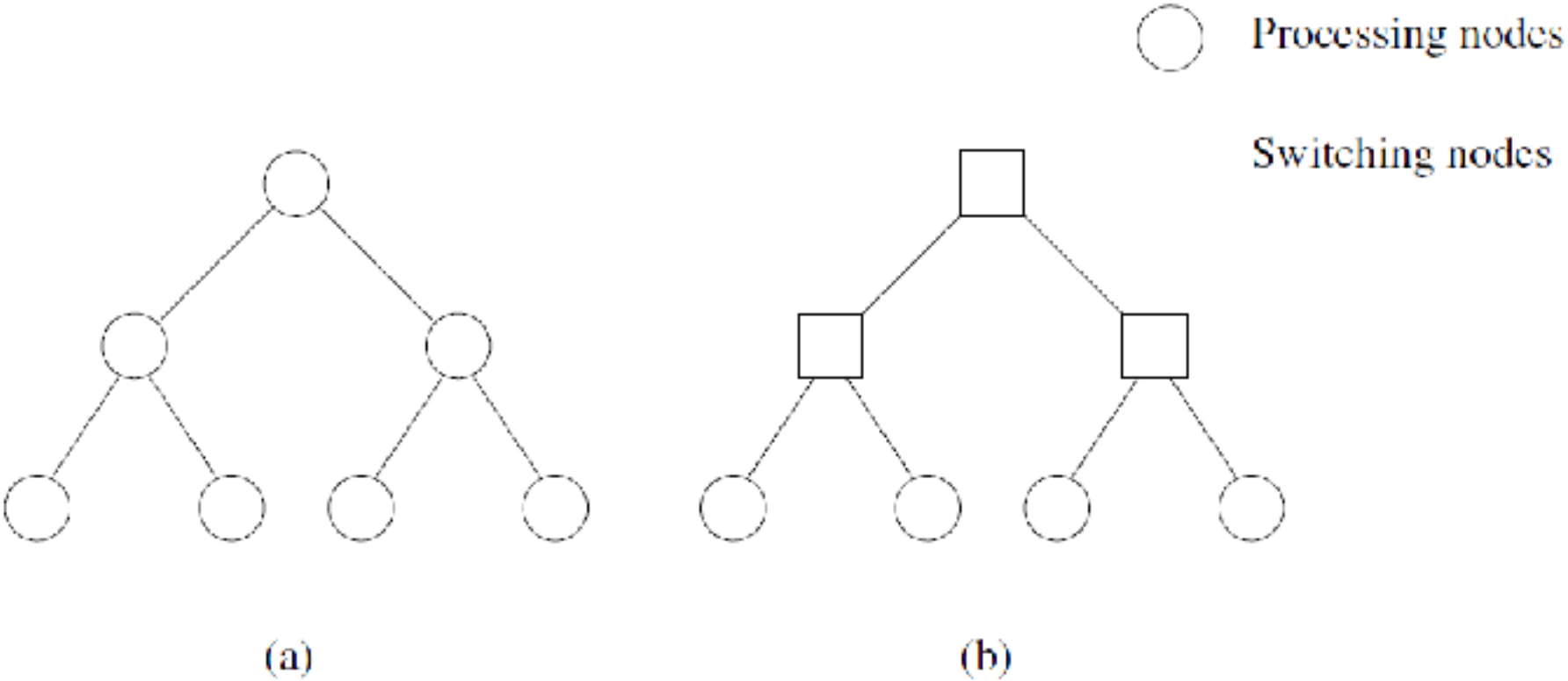




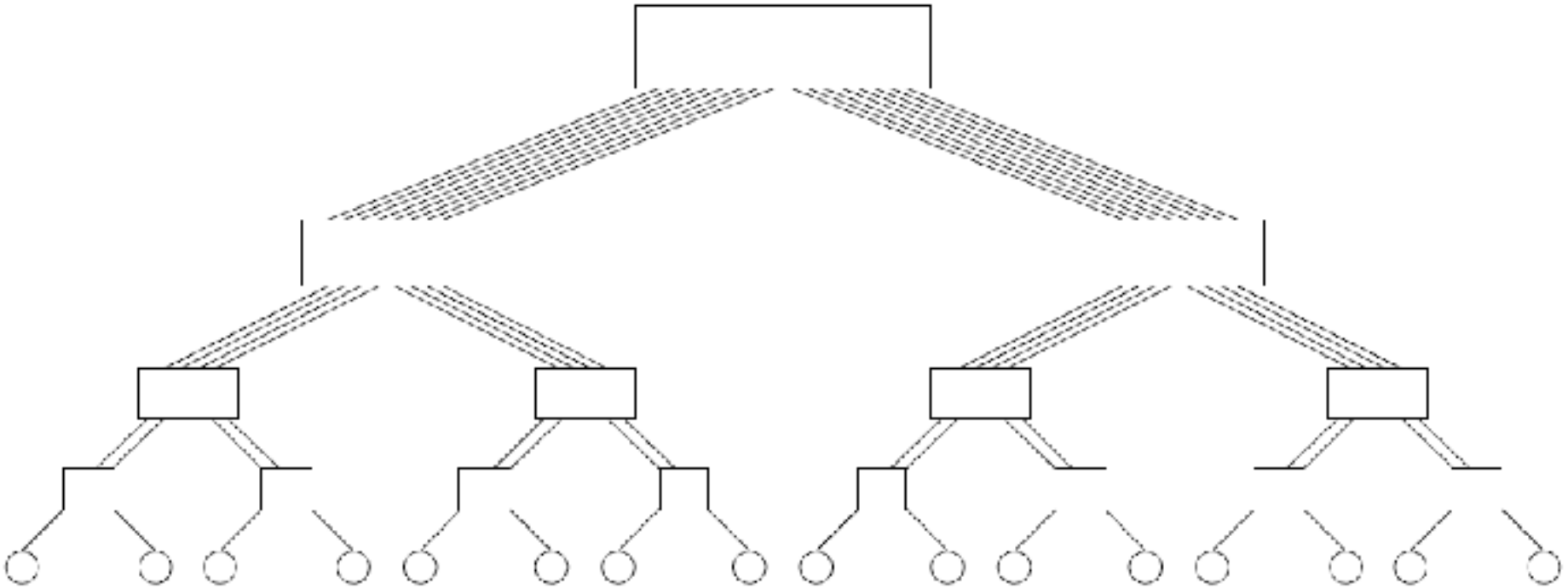
Fyzická organizácia paralelných systémov

- Stromová topológia
- Vzdialenosť medzi uzlami je max. 2 log p
- Prepojenia na vyšších úrovniach – väčšia prevádzka
- Fat-tree – prepojenia mohutnú s rastúcou úrovňou

Fyzická organizácia paralelných systémov



Fyzická organizácia paralelných systémov



Fyzická organizácia paralelných systémov

- **Metriky statických sietí**
- Polomer – vzdialenosť medzi najvzdialenejšími uzlami
- Šírka bisekcie – minimálny počet prepojení, ktoré je potrebné odstrániť, aby sa vytvorili dve rovnaké siete
- Cena – počet prepojení alebo prepínačov, či iné faktory ovplyvňujúce cenu

Fyzická organizácia paralelných systémov

Network	Diameter	BisectionWidth	Arc Connectivity	Cost (No. of links)
Completely-connected	1	$p^2/4$	$p-1$	$p(p-1)/2$
Star	2	1	1	$p-1$
Complete binary tree	$2\log((p-1)/2)$	1	1	$p-1$
Linear array	$p-1$	1	1	$p-1$
2-D mesh, no wraparound	$2(\sqrt{p}-1)$	$\sqrt{p}$	2	$2(p-\sqrt{p})$
2-D wraparound mesh	$2\lfloor\sqrt{p}/2\rfloor$	$2\sqrt{p}$	4	$2p$
Hypercube	$\log p$	$p/2$	$\log p$	$(p\log p)/2$
Wraparound *k*-ary *d*-cube	$d\ \lfloor k/2 \rfloor$	$2k^{d-1}$	$2d$	dp

Fyzická organizácia paralelných systémov

- **Metriky dynamických sietí**
  - Metriky vychádzajú z metrík pre statické siete
  - Prepínač spôsobuje oneskorenie – považovaný za uzol
  - Priemer – max. vzdialenosť medzi nejakými dvoma uzlami, indikuje to max. oneskorenie v sieti (iba procesory ale aproximatívne uvažujeme všetky)
  - Konektivita – minimálny počet uzlov nevyhnutných na rozdelenie siete na dve časti (iba prepínače ale aproximatívne uvažujeme všetky)
  - Hranová konektivita – počet hrán, ktorých odstránenie spôsobí rozdelenie siete na dve oddelené siete
  - Šírka bisekcie – minimálny počet prepojení, ktoré je potrebné odstrániť, aby sa vytvorili dve siete s rovnakým počtom procesorov

Fyzická organizácia paralelných systémov

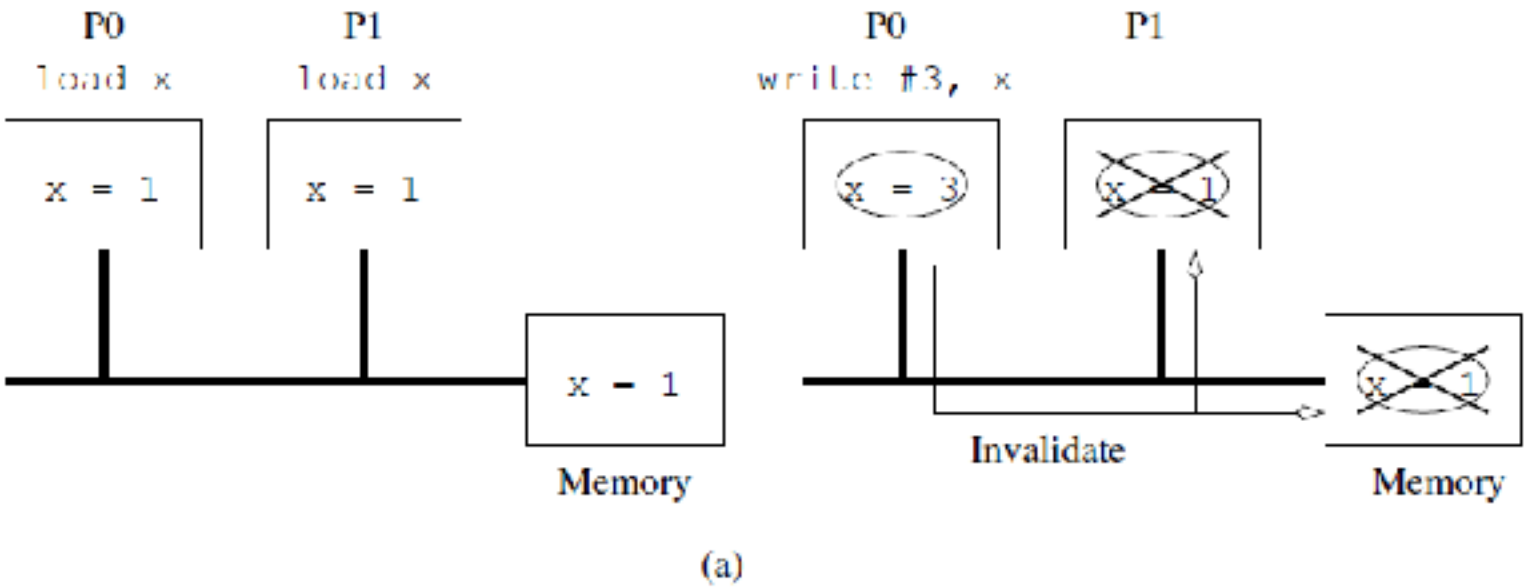
Network	Diameter	Bisection Width	Arc Connectivity	Cost (No. of links)
Crossbar	1	p	1	p^2
Omega Network	$\log p$	$p/2$	2	$p/2$
Dynamic Tree	$2 \log p$	1	2	$p - 1$

Fyzická organizácia paralelných systémov

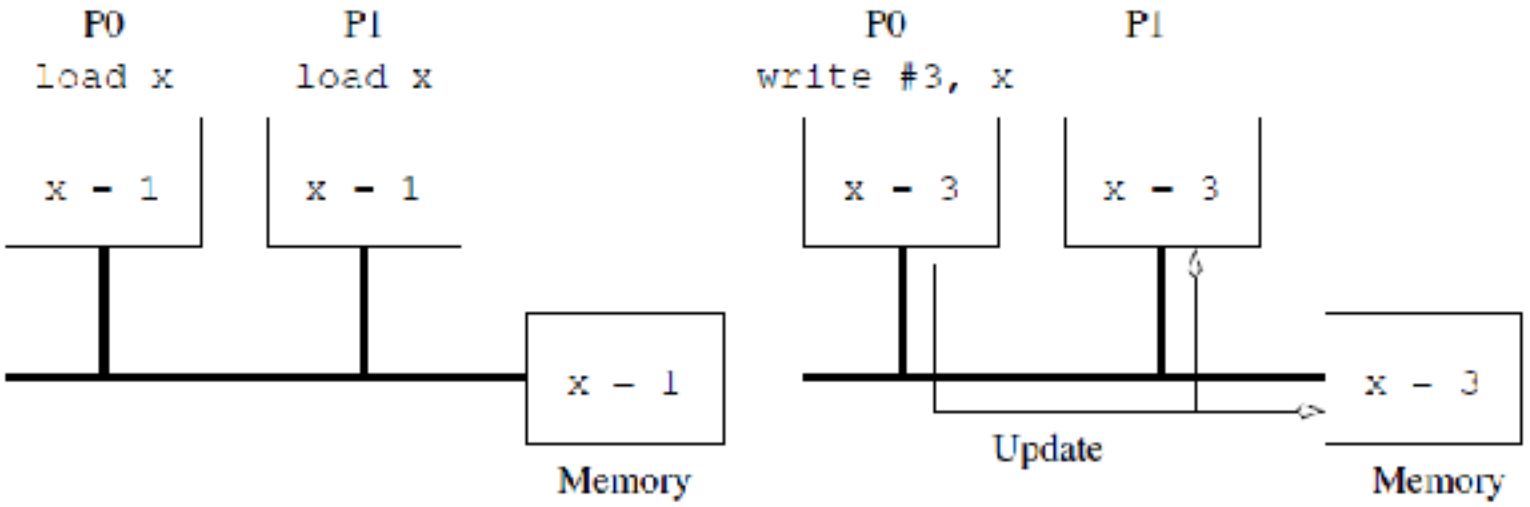
- **Koherencia vyrovnávacej pamäte v multiprocesorových systémoch**
- Prepojovacie siete – mechanizmus na komunikáciu údajov
- Systémy so zdieľanou pamäťou – ďalší HW na zabezpečenie konzistencie údajov existujúcich vo viacerých kópiách vo vyrovnávacích pamätiach

- Invalidate vs. update

Fyzická organizácia paralelných systémov



(a)



(b)

Fyzická organizácia paralelných systémov

- Update protokol
  - Hodnota načítaná procesorom, a už nepotrebná – veľká nadbytočnosť pri jej aktualizácií
  - Vhodné pri poprekladanom testovaní a zápise
- Update aj invalidate
  - Nadbytočnosť spojená s „nepravým“ zdieľaním (false sharing), dva údaje nie sú zdieľané, ale sa nachádzajú v rovnakom bloku vo vyrovnávacej pamäti

Cena komunikácie v paralelných systémoch

- Komunikácia – významný zdroj nadbytočnej réžie v paralelných systémoch
- Cena komunikácie závisí od viacerých okolností
  - Programátorského modelu
  - Topológie siete
  - Spracovania a smerovania dát
  - Protokolov

Cena komunikácie v paralelných systémoch

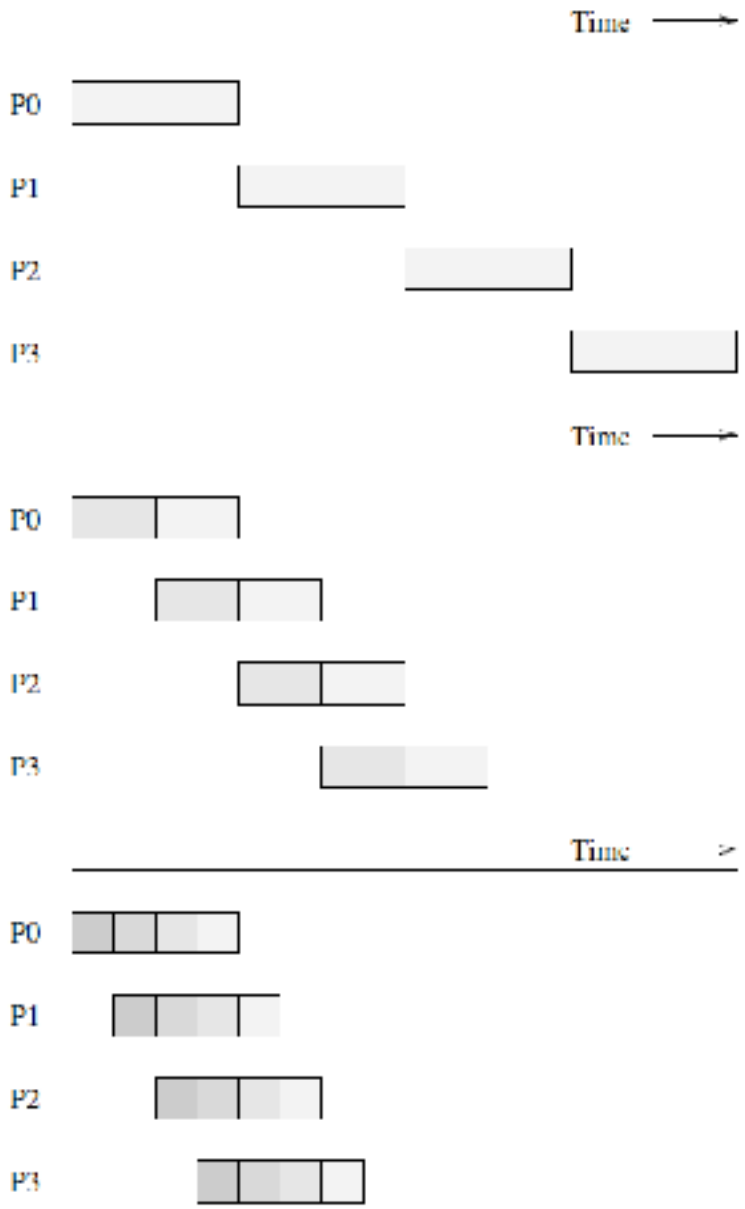
- Prenos správy cez komunikačnú sieť
  - *Startup time* (t_s): Čas strávený na zasielajúcich a príjmajúcich uzloch, programovanie smerovačov, atď.
  - *Per-hop time* (t_h): Čas závislý od počtu „skokov“ ovplyvnený faktormi ako oneskorenie na prepínačoch, na sieti, atď.
  - *Per-word transfer time* (t_w): Čas závislý od dĺžky správy, závisí od šírky pásma, protokoloch detekcie a opravy chýb, atď.

Cena komunikácie v paralelných systémoch

- Store and forward routing (ulož a prepošli smerovanie)
  - Správa smerovaná cez viaceré uzly je najskôr uzlom celá prijatá a potom preposlaná ďalej
  - Cena komunikácie pre správu o dĺžke m cez l sieťových prepojení:
  - $t_{comm} = t_s + (m\ t_w + t_h)\ l$

  - t_h väčšinou malé a vzťah môže byť aproximovaný:
  - $t_{comm} = t_s + m\ t_w\ l$

Cena komunikácie v paralelných systémoch



(a) A single message sent over a store-and-forward network

(b) The same message broken into two parts and sent over the network.

(c) The same message broken into four parts and sent over the network.

Cena komunikácie v paralelných systémoch

- Packet routing (paketové smerovanie)
  - Store-and-forward – slabé využívanie prostriedkov
  - Paketové smerovanie – rozloženie správy na menšie časti, pakety, a tie „prúdovým“ spôsobom preniesť
  - Každý paket môže cestovať inou trasou, musí teda obsahovať informáciu o smerovaní, potreba detekcie chýb, vytvárania postupností paketov, + iné informácie do hlavičky

  - Aproximácia ceny paketového smerovania:
  - $t_{comm} = t_s + t_h\, l + t_w\, m$

  - Člen t_w reprezentuje réžiu spojenú so smerovaním a spracovaním hlavičiek paketov

Cena komunikácie v paralelných systémoch

- Cut – Through routing
  - Do extrému prevedené paketové smerovanie
  - Základné jednotky malé pakety flits
  - Minimálna informácia do hlavičky – smerovanie flits cez rovnakú cestu v rámci sekvencie paketov
  - Detekcia chýb vykonaná až nad celou správou
  - Nie je potrebné číslovanie flits

  - Aproximácia ceny cut-through smerovania:
  - $t_{comm} = t_s + t_h\, l + t_w\, m$

  - Rovnaká, ako u paketového smerovania, ale t_w je menšie

Cena komunikácie v paralelných systémoch

- Cena komunikácie medzi dvoma uzlami vzdialenými l skokov je daná:
- $t_{comm} = t_s + t_h\, l + t_w\, m$

- Vo vzťahu t_h je typicky oveľa menšie ako t_s a t_w
- Člen $t_h\, l$ je možné zanedbať, hlavne ak m je veľké
- Tiež je často nemožné riadiť smerovanie a umiestňovanie úloh

- Aproximácia ceny komunikácie je teda:
- $t_{comm} = t_s + t_w\, m$

Cena komunikácie v paralelných systémoch

- Výraz pre cenu platí iba pre nezahltené siete
- Ak je prepojenie použité pre odkomunikovanie viacerých správ, člen t_w musí byť zodpovedajúco zmenený
- Rôzne spôsoby využitia komunikačnej siete zahlcujú rôzne komunikačné siete rôznym spôsobom
- Potrebné zohľadniť pri komunikácií

Cena komunikácie v paralelných systémoch

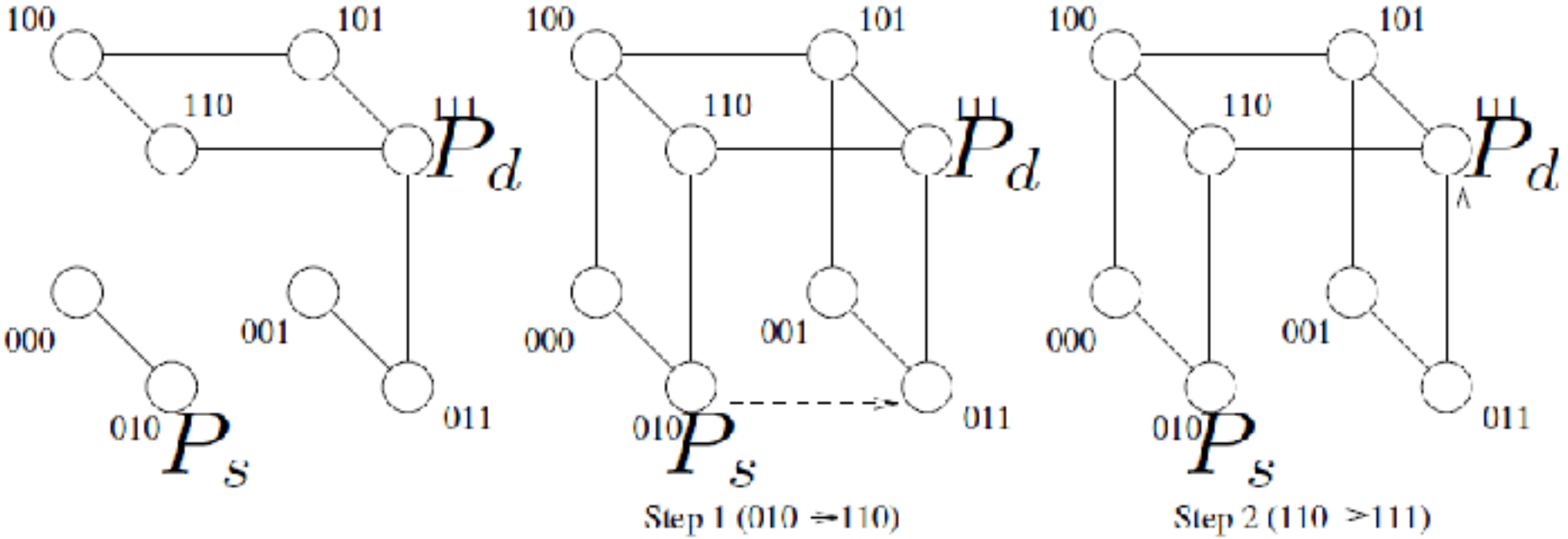
- Systémy so zdieľanou pamäťou
- Potreba zohľadniť ďalšie faktory
- Konečná veľkosť vyrovnávacích pamätí – cache trashing
- Réžia spojená s invalidovaním a upravovaním (cache coherency) vyrovnávacej pamäte sa ťažko predikujú
- Zložité modelovanie priestorovej lokálnosti
- Skoré načítanie (prefetching) môže byť významné pri redukcií réžie spojenej s prístupom k údajom
- Zložité modelovanie s „nepravého“ zdieľania (false sharing) a súťaženia o zdroje (contention)

Smerovanie v komunikačných sieťach

- Ako určiť cestu pre smerovanie údajov od zdroja k cieľu?
  - Zabránenie uviaznutiu – usporiadanie podľa dimenzií alebo tzv. e-cube smerovanie
  - Nevytvárať „horúce-body“ – dvojkrokové smerovanie je často používané, správa zo zdroja *s* do cieľa *d* je preposlaná cez náhodne vybraný procesor *i*

Smerovanie v komunikačných sieťach

- Smerovanie správy z uzla P_s (010) do uzla P_d (111) v trojdimenzionálnej hyperkocke pomocou E-cube smerovania



Techniky mapovania

- Mapovanie známej komunikačnej štruktúry na danú prepojovaciu topológiu
- Algoritmus navrhnutý pre jednu komunikačnú sieť prenášame na druhú komunikačnú sieť s inou topológiou

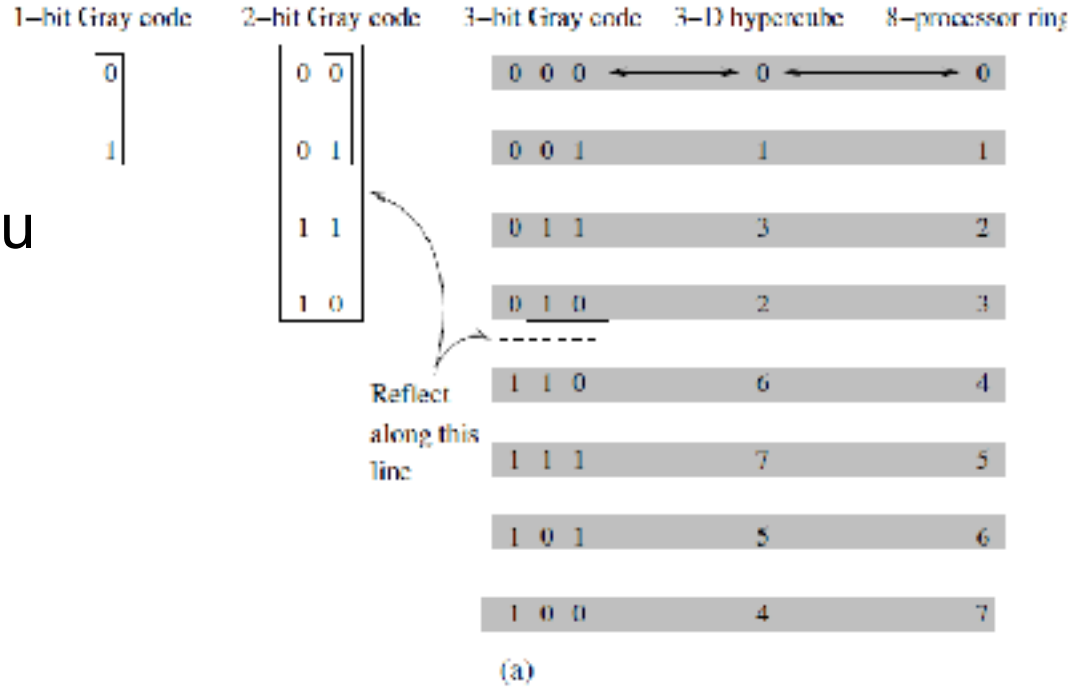
- Mapovanie medzi grafmi

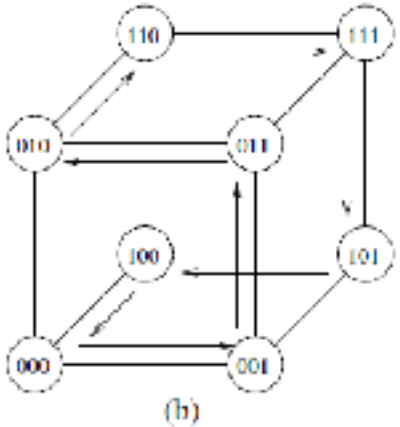
Techniky mapovania

- Mapovanie grafu *G(V,E)* na graf *G'(V',E')*
- Dôležité metriky:
- Congestion (nasýtenie) – max. počet hrán z E mapovaných na E'
- Dilation (dilatácia) – maximálny počet prepojení v E', na ktoré sa mapuje nejaká hrana z E
- Expansion (expanzia) – pomer uzlov vo V' k uzlom vo V

Techniky mapovania

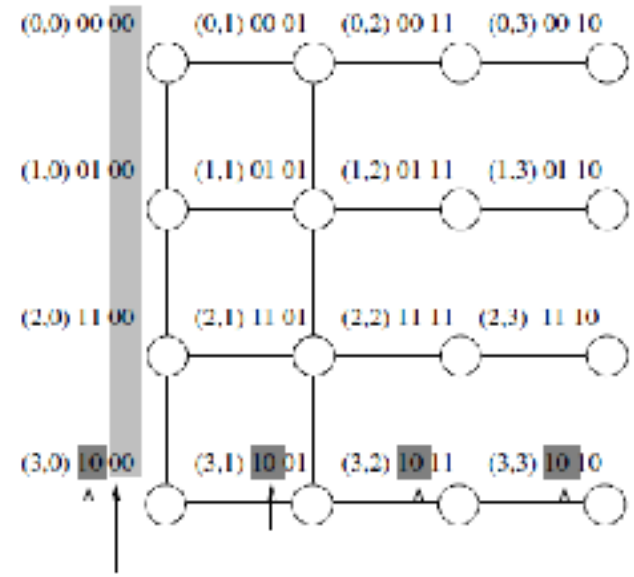
- 1D graf na hyperkocku
  - Pomocou Grayovho kódovania

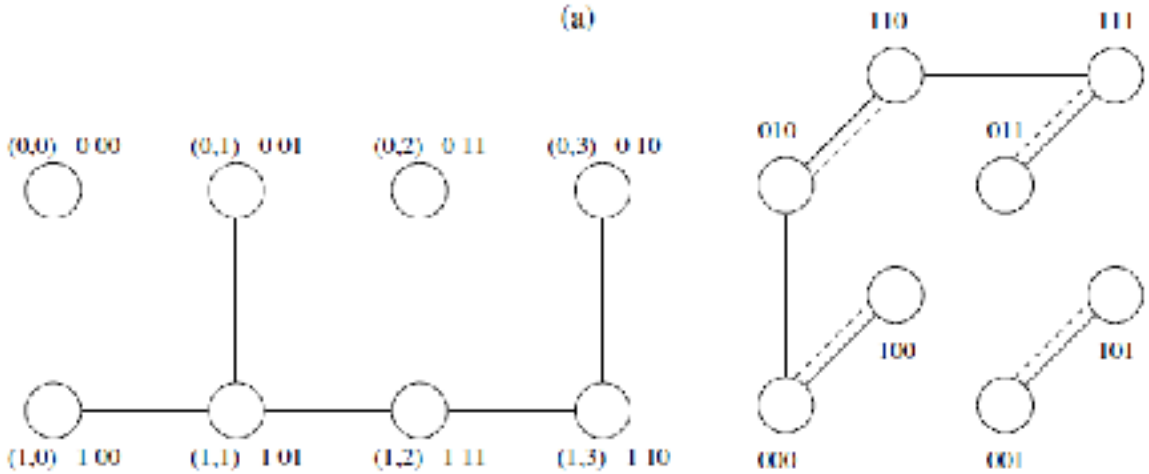




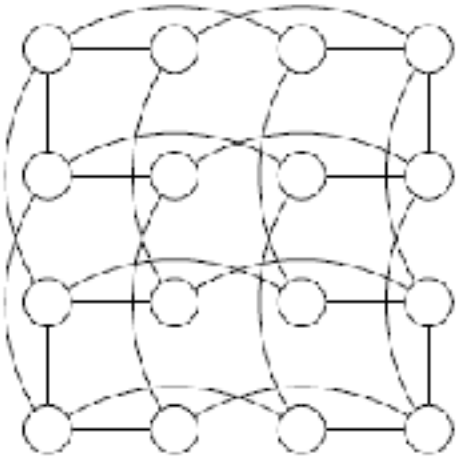
Techniky mapovania

- 2D mesh na hyperkocku
  - Pomocou Grayovho kódovania



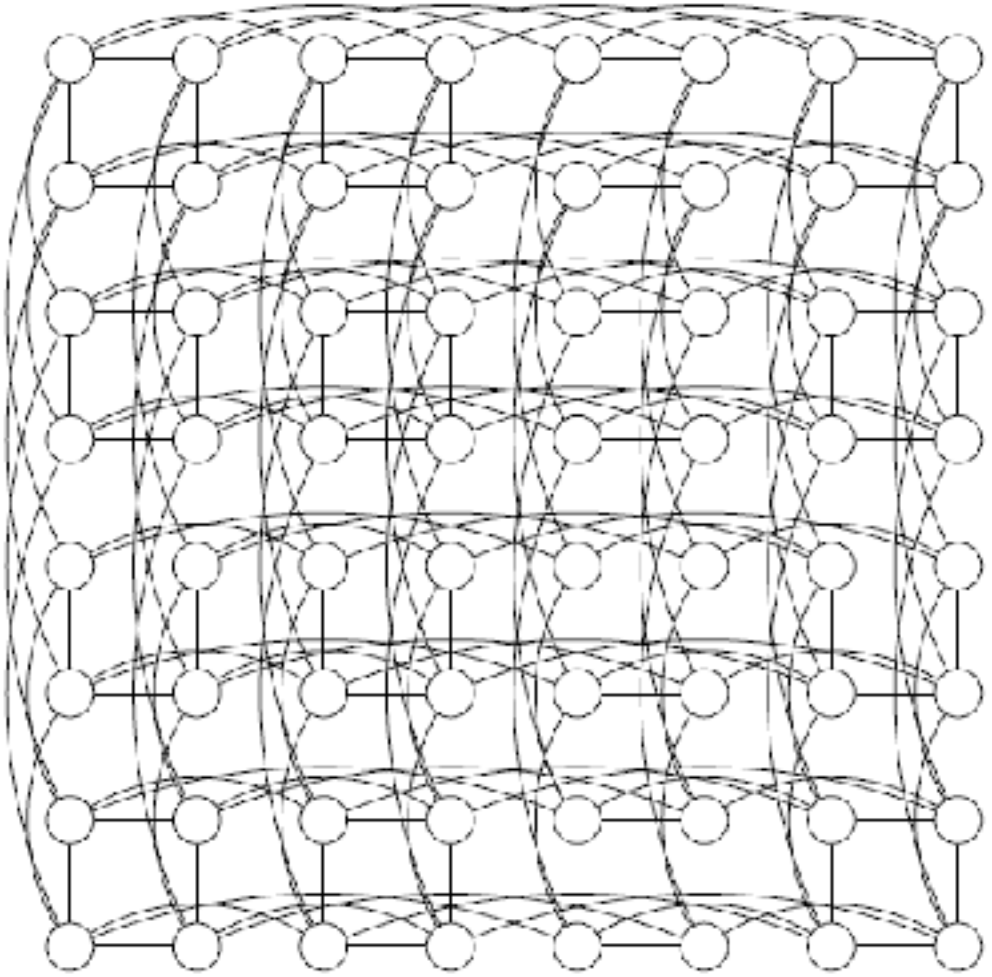


Techniky mapovania



(a) $P = 16$

- Hyperkocka na 2D mesh



(b) $P = 32$

Zdroje

- Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar. Introduction to ParallelComputing, 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 ParallelComputing, 2nd Edition, Addison-Wesley 2003„Introduction to Parallel Computing“ http://www-users.cs.umn.edu/~karypis/parbook/