IA039
Paralelní počítače

Paralelní počítače

- Small-scale multiprocessing
  - 2–16 procesorů
  - převážně SMP (sdílená paměť)
- Large-scale multiprocessing
  - > 100 (i tisíce) procesorů
  - Zpravidla distribuovaná paměť

Paralelní počítače (II)

- Architektura
  - Single Instruction Multiple Data, SIMD
  - Multiple Instruction Multiple Data, MIMD
- Programovací modely
  - Single Program Multiple Data, SMPD
  - Multiple programs Multiple Data, MPMD

Architektura – SIMD

- Procesory synchronizovány
  - Všechny vykonávají vždy stejnou instrukci
  - Analogie vektorových procesorů
- Jednoduché procesory
- Jednodušší programovací model

Architektura – MIMD

- Plně asynchronní systém
- Procesory zcela samostatné
  - Není třeba speciální výroba (off-the-shelf)
- Výhody
  - Vyšší flexibilita
  - Teoreticky vyšší efektivita
- Nevýhody
  - Explicitní synchronizace
  - Složité programování

Komunikační modely

- Sdílená paměť (Shared Memory Architecture)
- Předávání zpráv (Message passing)

Sdílená paměť

- Paměť odělená od procesorů
- Uniformní přístup k paměti
- Nejsnazší propojení – sběrnice
- „Levná“ komunikace
- Složité prokládání výpočtu a komunikace (aktivní čekání)

Předávání zpráv

- Každý procesor „viditelný“
- Vlastní paměť u každého procesoru
- Explicitní komunikace – předávání zpráv
- Vysoká cena komunikace (výměny dat)
- Možnost prokládání výpočtů a komunikace

Hybridní systémy

- Nonuniform memory access architecture (NUMA)
- Cache-only memory access architecture (COMA)
- Distributed shared-memory (DSM)

Non-uniform memory access

- Přístup k ruzným fyzickým adresám trvá různou dobu
- Umožňuje vyšší škálovatelnost
- Potenciálně nižší propostnost
- Koherence vyrovnávacích pamětí
  - ccNUMA

Cache only memory access

- NUMA s charakterem vyrovnávací paměti
- Data putují k procesorům, které je používají
- Pouze zdánlivá hierarchie
  - Systém musí hlídat, že má jedinou kopii
- Experimentální

Distributed shared-memory

- Distribuovaný systém – cluster
  - Lokální paměť každého uzlu
  - Vzdálená paměť ostatních uzlů

  „Fikce“ jedné rozsáhlé paměti
- Hardwarové řešení
  - Zpravidla využívá principů virtuální paměti
  - Transparentní
- Softwarové řešení
  - Knihovna
  - Netransparentní, progamátor program musí explicitně přizpůsobit

Koherence vyrovnávacích pamětí

- Příčiny výpadku vyrovnávací paměti:
  - **Compulsory** miss: 1. přístup k datům
  - **Capacity** miss: nedostatečná kapacita
  - **Conflict** miss: různé adresy mapovány do stejného místa
  - **Coherence** miss: různá data v různých vyrovnávacích pamětích
- Poslední případ se týká multiprocesorů

Řešení problému koherence

- Vyrovávací paměti musí vědět o změně
- Metody založené na broadcastu
- Adresářové metody

Snoopy cache

- Broadcastový přístup
  - Propojovací sítě s „přirozeným“ brodcastem – sběrnice
- Každý procesor sleduje *všechny* přístupy k paměti

Zneplatnění

- Reakce na změnu dat ve vzdálené (vyrovnávací) paměti
- Řádka v aktuální (naslouchající) vyrovnávací paměti je zneplatněna
- V případě opětného přístupu je přehrána ze vzdálené paměti

Update

- Řádka je okamžitě obnovena
- Při opětovném přístupu je již k dispozici
- Nevýhody
  - Falešné sdílení (nepracují na stejných datech)
  - Přílišné zatížení sběrnice
- Nelze rozhodnout, zda update nebo zneplatnění je obecně lepší

Rozšiřitelnost (Scalability)

- Není jednotná definice
- Používané základní formulace – rozšiřitelný je takový systénm, pro nějž platí:
  - Výkon roste lineárně s cenou
  - Je zachován konstantní poměr Cena/Výkon
- Alternativní parametr – **Míra rozšiřitelnosti**
  - Změna výkonu přidáním procesoru
  - Změna ceny přidáním procesoru
  - Smysluplný rozsah počtu procesorů

Zrychlení

$$S(N) = \frac{T_{EXEC}(1)}{T_{EXEC(N)}} = \frac{T_{comp}(1) + T_{comm}(1)}{T_{comp}(N) + T_{comm}(N)}$$

- Ideální zrychlení vyžaduje

$$T_{comp}(N) = T_{comp}(1)/N$$
$$T_{comm}(N) = T_{comm}(1)/N$$

Zrychlení – komentář

- Teoretický pojem, realita závisí na aplikaci
  - Různé hodnoty pro různé aplikace
  - Vliv paralelizovatelnosti problému (Amdalův zákon)

Rozšiřitelné propojovací sítě

- Požadavky na ideální síť:
  - Nízka cena rostoucí lineárně s počtem procesorů (N)
  - Minimální latence nezávislá na N
  - Propustnost rostoucí lineárně s N

Vlastnosti sítí

- Tři základní komponenty
  - Topologie
  - Přepínání dat (jak se data pohybují mezi uzly)
  - Směrování dat (jak se hledá cesta mezi uzly)

Propojovací sítě

- Rozlišujeme následující základní parametry
  - Velikost sítě – počet uzlů N
  - Stupeň uzlu d
  - Poloměr sítě D
    - Nejdelší nejkratší cesta
  - Bisection width B
  - Redundance sítě A
    - Minimální počet hran, které je třeba odstranit, aby se síť rozpadla na dvě
  - Cena C
    - Počet komunikačních linek v síti

Bisection width

- Šířka rozpůlení
  - Minimální počet linek, které je třeba odstranit, aby se systém rozpadl na dvě stejné části
- **Bisection bandwidth** – propustnost při rozpůlení
  - Celková kapacita (propustnost) výše odstraněných linek
- Ideální vlastnost:
  - Bisection badnwidth vztažená na procesor je v daném systému konstantní.

Topologie přepínacích sítí

- Klasifikace na základě rozměru
  - Jednorozměrné
  - Dvourozměrné
  - Třírozměrné
  - Hyperkrychle

Jednorozměrné propojovací sítě

- Linerární pole
- Jednotlivé prvky navázány na sebe
  - „Korálky“
- Nejjednodušší
- Nejhorší vlastnosti

Dvourozměrné propojovací sítě

- Kruh
  - Uzavřené lineární pole
- Hvězda
- Strom
  - Snižuje poloměr sítě ($2 \log \frac{N+1}{2}$)
  - Stále špatná redundance a bisection (band)width
  - Tlustý strom (fat tree)
    - Přidává redundantní cesty ve vyšších úrovních

Dvourozměrná mřížka

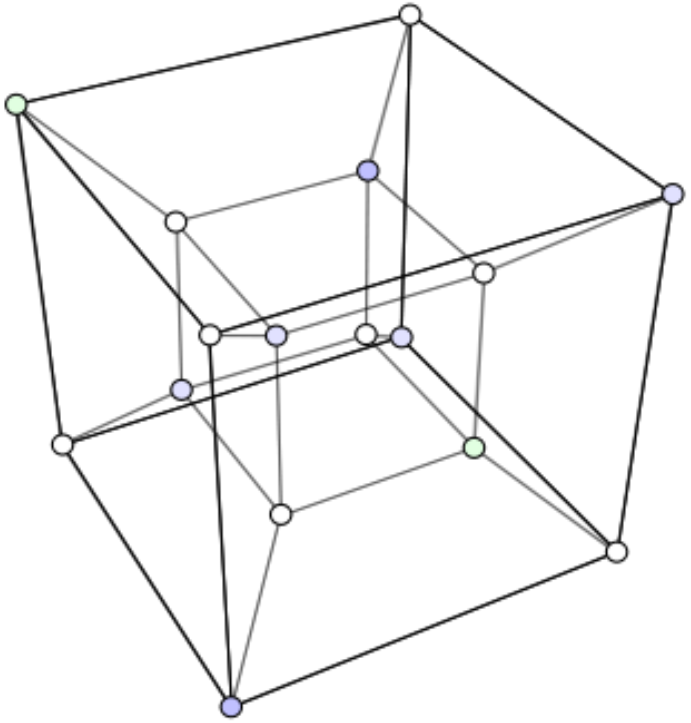
- Velmi populární
  - Dobré vlastnosti
    - Poloměr $2(N^{1/2} - 1)$
    - Bisection $N^{1/2}$
    - Redundance 2
  - Avšak vyšší cena a proměnný stupeň uzlu
- Torus
  - Uzavřená dvourozměrná mřížka
    - Poloměr $N^{1/2}$
    - Bisection $2N^{1/2}$
    - Redundance 4
    - Vyšší cena – přidá $2N^{1/2}$ hran

Třírozměrná síť

- Vlastnosti
  - Poloměr $3(N^{1/3} - 1)$
  - Bisection $N^{2/3}$
  - Redundance 3
  - Cena akceptovatelná $2(N - N^{2/3})$
- Konstrukčně složitá

Hyperkrychle

- Velmi zajímavá topologie
- Obecně n-rozměrná krychle
- Základní parametry
  - Poloměr $\log N$
  - Bisection $N/2$
  - Redundance $\log N$
  - Vyšší cena $(N \log N)/2$



- Mřížky speciálními případy hyperkrychle
- Snadné nalezení cesty
  - Binární číslování uzlů

Plně propojené sítě

- Teoretická konstrukce
- Vynikající poloměr (1)
- Neakceptovatelná cena $(N * (N-1)/2)$ a stupeň uzlu $(N-1)$

Přepínání 

- Konkrétní mechanismus, jak se paket dostane ze vstupu na výstup
- Základní přístupy
  - Přepínání paketů, store-and-forward
  - Přepínaní obvodů
  - Virtuální propojení (cut-through)
  - Směrování červí dírou (wormhole routing)

Store-and-forward

- Celý paket se uloží
- A následně přepošle
- Jednoduché (první generace paralelních počítačů)
- Vysoká latence $\frac{P}{B} * D$
  - P je délka paketu, B je propustnost a D je počet „hopů“ (vzdálenost)

Přepínání okruhů

- Tři fáze
  - Ustavení spojení – zahájeno vzorkem (probe)
  - Vlastní přenos
  - Zrušení spojení
- Výrazně nižší latence $\frac{P}{B} * D + \frac{M}{B}$
  - P je v tomto případě délka vzorku a M je délka zprávy (nejsou nutné pakety)
  - Pro $P << M$ latence není závislá na délce cesty

Virtuální propojení

- Zprávu rozdělíme do menších bloků – flow control digits (flits)
- Posíláme jednotlivé flits kontinuálně
  - Jsou-li buffery dostatečně velké, odpovídá přepínání okruhů
- Latence $\frac{HF}{B} * D + \frac{M}{B}$
  - HF je délka flitsu, zpravidla $HF << M$

Červí díra

- Speciální případ virtuálního propojení
- Buffery mají právě délku flits
- Latence nezávisí na vzdálenosti
  - Analogie pipeline
- Podporuje replikace paketů
  - Vhodné pro multicast a broadcast

Virtuální kanály

- Sdílení fyzických kanálů
- Několik bufferů nad stejným kanálem
  - Flits uložen v příslušném bufferu
- Využití
  - Přetížená spojení
  - Zábrana deadlocku
  - Mapování logické na fyzickou topologii
  - Garance propustnosti systémovým datům

Směrování v propojovacích sítích

- Hledání cesty
- Vlastnosti
  - Statické směrování
    - Zdrojové
    - Distribuované
  - Adaptivní směrování (vždy distribuované)
- Minimální a ne-minimální

Fault tolerance propojovacích sítí

- Kontrola chyb
- Potvrzování zpráv
- Opakované zasílání zpráv

Koherence vyrovnávacích pamětí II

- Snoopy schema založené na broadcastu
  - Nepoužitelné u složitějších propojovacích sítí
  - Není rozšiřitelné (scalable)
- Redukce „oslovených“ vyrovnávacích pamětí – Adresáře
  - Položka u každého bloku paměti
  - Odkazy na vyrovnávací paměti s kopií tohoto bloku
  - Označení exkluzivity (právo pro čtení)

Adresářové přístupy

- Tři základní schemata
  - Plně mapované adresáře
  - Částečně (Limited) mapované adresáře
  - Provázané (chained) adresáře
- Zhodnocení vlastností
  - Na základě velikosti potřebné paměti
  - Na základě složitosti (počtu příkazů) protokolu

Plně mapované adresáře

- Každá adresářová položka má tolik údajů, kolik je vyrovnávacích pamětí (procesorů)
- Bitový vektor „přítomnosti“
  - Nastavený bit znamená, že příslušná vyrovnávací data má kopii bloku paměti
- Příznak exkluzivity
  - Stačí jeden na blok
  - Jen jedna vyrovnávací paměť
- Příznaky v každé vyrovnávací paměti (každý blok)
  - Příznak platnosti
  - Příznak exkluzivity

Omezené adresáře

- Plné adresáře velmi paměťově náročné
  - Velikost paměti: P^2M/B
    - P je počet vyrovnávacích pamětí
    - M velikost hlavní paměti
    - B velikost bloku
- Data nejsou zpravidla široce sdílena
  - Většina adresářových bitů má hodnotu nula
- Použití přímých odkazů
  - Nebude již stačit jeden bit

Omezené adresáře II

- Množina ukazatelů na vyrovnávací paměti
  - Dynamická alokace dle potřeby
- Vlastnosti
  - Počet bitů ukazatele: $\log_2 P$
  - Počet položek v poolu ukazatelů: k
  - Výhodnější než přímo mapovaná: pokud $k < \frac{P}{\log_2 P}$
- Informovány jen explicitně uvedené vyrovnávací paměti

Přetečení

- Pokud přestanou stačit položky
  - Příliš mnoho sdílených bloků
- Možné reakce
  - Zneplatnění všech sdílených (brodcast, Dir$_i$B)
  - Výběr jedné položky (i náhodně) a její zneplatnění (bez broadcastu, Dir$_i$NB)

Další schemata

- Coarse-vector (Dir$_i$CV$_r$)
  - r je velikost regionu (více procesorů), kterému odpovídá jeden bit (tedy více procesorů)
  - Přepnutí interpretace při přetečení
    - Omezený broadcast všem procesorům v oblasti.
- LimitLESS: programové přerušení při přetečení

Provázaná schemata

- Cache-Based linked-list
- Centrálně pouze jediný ukazatel
- Ostatní ukazatele svázány s vyrovnávací pamětí
  - Vyrovnávací paměti „provázany“ ukazateli
- Výhody
  - Minimalizace paměťových nároků
- Nevýhody:
  - Složitý protokol.
  - Zvýšená komunikace (více zpráv než nutno)
  - Zápis je delší (sekvenční procházení seznamu)

Hierarchické adresáře

- Použité v systémech s vícenásobnými sběrnicemi
- Hierarchie vyrovnávacích pamětí
  - Vyšší úroveň na každém propojení sběrnic
  - Vyšší paměťové nároky
    - Vyšší úroveň musí držet kopie paměťových bloků sdílených nižší úrovní
    - Není třeba rychlá paměť
- V principu hierarchie snoopy protokolů

Zpoždění paměti 

- Paměť výrazně pomalejší než procesor
  - Čekání na paměť podstatně snižuje výkon systému
- Možná řešení:
- *Snížením zpoždění* – zrychlení přístupu
- *Ukrytím zpoždění* – překryv přístupu a výpočtu

Snížení zpoždění paměti

- **NUMA:** NonUniform Memory Access
  - Každé logické adrese odpovídá konkrétní fyzická adresa
- **COMA:** Cache-Only Memory Architecture
  - Hlavní paměť je chápána jako *attraction memory*.
  - Řádky paměti se mohou volně přesouvat.
  - Mohou existovat sdílené kopie řádků paměti.

Rekapitulace

Communication to computation ratio	NUMA		COMA	
	Small working set	Large working set	Small working set	Large working set
Low	Good	Medium	Good	Good
High	Medium	Poor	Poor	Poor

Ukrytí zpoždění paměti

- Modely slabé konzistence
- Prefetch
- Procesory s vícenásobnými kontexty
- Komunikace iniciovaná producentem

Slabá konzistence

- Nepožaduje striktní uspořádání přístupů ke sdíleným proměným vyjma synchronizačních.
- **Release consistency:**
  - Zavedení operací *acquire* a *release*
- *Fence* operace
  - Vynucené dokončení rozpracovaných operací

Prefetch

- Přesun dat k procesoru s předstihem.
  - *Binding* prefetch
    - Data přesunuta až k procesoru
    - Možné porušení konzistence
  - *Nonbinding* prefetch
    - Data přesunuta pouze do vyrovnávací paměti
- HW Prefetch
- SW Prefetch
- Speciální instrukce *prefetch-exclusive*: read následovaný příkazem write.

Procesory s vícenásobnými kontexty

- Podpora multitherading
- Vyžaduje
  - Velmi rychlé přepnutí kontextu
  - Vysoký počet registrů
- Řada experimentálních systémů
  - HEP (70. léta)
  - Tera
  - *T

Komunikace iniciovaná producentem

- Analogie invalidace a update při cache koherenci
- Specifické využití pro message-passing (Cray T3D) nebo block-copy (počítače se sdílenou pamětí).
- Vhodné např. pro přesun velkých bloků dat či pro synchronizaci zámky (locks).

Podpora synchronizace

- Synchronizace tvoří „horká místa“
- Základní synchronizační primitivy:
  - Vzájemné vyloučení
  - Dynamické rozložení zátěže
  - Informace o událostech
  - Globální serializace (bariéry)

Vzájemné vyloučení

- K dané proměnné má v daném okamžiku přístup nejvýše jeden proces
- Univerzální, ovšem zpravidla zbytečně drahé
- Synchronizační konstrukce vyšších jazyků
  - Semafory
  - Monitory
  - Kritické oblasti
- Základem – hardwarová podpora
  - `test&set` instrukce
  - `test-and-test&set` instrukce

Spin waiting protocol

test&set

- Vlastnosti

  char *lock;

  while (exchange(lock, CLOSED) == CLOSED);

- Busy waiting
- Vysoké požadavky na přenos (časté zneplatnění) u multiprocerů

test-and-test&set

- Vlastnosti

  ```
  for (;;)
  while (*lock == CLOSED);
  if (exchange(lock, CLOSED) != CLOSED)
  break;
  ```

- Využití vyrovnávacích pamětí
  - první testy nad sdílenou kopií

Použití front

- Výhodnější *Collision avoidance schemata*
- Queue on lock bit (QOLB) protokol
- Nejefektivnější implementace
- Procesy řazeny do fronty
  - Po uvolnění zámku aktivován proces v čele fronty
  - Není třeba žádný sdílený přenos dat

Zámky v multiprocesorech

- Souvisí i s možností dynamického rozložení zátěže
- Využití čitače s atomickou operací
  - Fetch&Op – čitače, např. Op==Add

```
fetch&add ( x, a)
int *x, a;
{ int temp;
  temp = *x;
  *x += a;
  return (temp);
}
```

  - Compare&Swap – seznamy

Použití

Informace o (globálních) událostech používána především producentem jako prostředek, kterým jsou konzumenti informováni o nově dostupných datech, a dále při informaci o globální změně ve skupině ekvivalentních procesů (změna určitého stavu, která musí být oznámena všem procesům).