# Přednáška #12: **Úvod do paralelních počítačů**

## Paralelní počítače a architektury

**Definice 1.** *(Almasi, Gottlieb 89) Paralelní (//) počítač je skupina* **výpočetních uzlů** *(procesních prvků, angl. PE), které spolu komunikují a spolupracují tak, aby vyřešily velké problémy rychle(ji). PE = procesor + paměť.* ♣

- **Problémy:**

(1) Návrh // počítačů (počet a granularita PE, způsob propojení, škálovatelnost.)
(2) Návrh efektivních // algoritmů (Je // čas $\times$ počet PE $\doteq$ složitost nejlepšího sekvenčního algoritmu?)
(3) Metody vyhodnocování kvality // algoritmů (Existence rychlých // řešení? Efektivnost využití procesorů?)
(4) // programovací jazyky (Jazykové konstrukce pro distribuci dat mezi PE, pro // provádění operací a příkazů a pro výměny dat.)
(5) // programovací nástroje (Trasovací a ladící nástroje, vizualizační nástroje, výkonnostní monitory.)
(6) Přenositelnost (// programy by měly být nezávislé na architektuře & efektivní!!!!)
(7) Paralelizující kompilátory a techniky (automatická paralelizace sekv. kódů.)
(8) . . .

# Význam // počítačů a // počítání?



**Hypotéza:** // počítače hrají a budou hrát stále významnější roli v životě naší civilizace.

**Důvody:**

(1) Nejpřirozenější způsob, jak zvýšit výkon či paměťovou kapacitu za hranice 1 procesoru, je spojení více procesorů dohromady (mikroprocesor je součástka).

(2) Křemíkové VLSI technologie se zdají přibližovat svým zásadním fyzickým limitům daným rychlostí světla a energetickými poměry mikrosvěta.

(3) Navíc nárůst výkonu cestou ILP mikroprocesorů má své meze (extrémně složité procesory pro IPC $\doteq$ 2).

(4) Současně ale pokroky VLSI technologií vedou k explozivnímu nárůstu výkonu a schopností procesorů díky vyšším hodinovým frekvencím & většímu počtu tranzistorů na čipu
$\Longrightarrow$ více procesorů (PE) lze integrovat na 1 čip.

(5) Hlavní překážka rozšíření // počítačů – SW (// jazyky, knihovny, algoritmy, serverové a vestavěné aplikace) – se pomalu ale jistě vyvíjejí a mohutní.

(6) Existují **kladné** zpětné vazby mezi pokroky v počítačových architekturách a technologiích na jedné straně a složitostí problémů ve vědecké a inženýrské oblasti na druhé straně:
Jinými slovy: Velké vědecké pokroky generují potřebu **řešit** nové velké a složité problémy
$\Longrightarrow$ je potřeba vyvíjet nové a stále výkonnější počítače pro jejich řešení
$\Longrightarrow$ po jejich vyřešení se smyčka opakuje.

# Velké problémy lidstva (Grand challenges)



Základní problémy vědy a inženýrství s širokým ekonomickými a vědeckými dopady na naší civilizaci vyžadují vysoce náročné počítání (`High Performance Computing (HPC)`):

- předpovídání počasí, modelování atmosféry,
- vědy o Zemi (modelováni moří a zemětřesení, ekologické vědy),
- zpracování obrovského množství dat ze satelitů,
- modelování v astronomii a kosmologii,
- biochemie, biotechnologie a genetika
  (nejvýkonnější počítač na světě IBM BlueGene),
- návrhy a testy složitých inženýrských systémů (letadla, auta, elektrárny),
- počítačové zkoušky jaderných zbraní a skladovaní jaderného odpadu,
- rozpoznání lidské řeči, modelováni lidské inteligence, robotika,
- dobývání znalostí a hledání na internetu
  (za Googlem stojí systém cca 100 000 výkonných počítačů),
- . . . a mnoho dalších.

# Příklad náročné aplikace: předpověď počasí



- Uvažujme předpověď počasí v **oblasti** o velikosti $3000 \times 3000 \times 11\ km^3$ na dobu **2 dnů**.
- Tato oblast je rozdělena na **segmenty** (např. metodou konečných prvků) o velikosti $0.1 \times 0.1 \times 0.1\ km^3 \implies$ počet segmentů je řádově $10^{11}$.
- Parametry modelu (teplota, rychlost větru) jsou počítány s časovým krokem $30$ minut.
- **Nové** hodnoty parametrů jednoho segmentu jsou počítány z **předchozích** hodnot parametrů tohoto segmentu a segmentů sousedních.
- Předpokládejme, že výpočet parametrů 1 segmentu spotřebuje $100$ instrukcí.
- Pak 1 **iterace** = **aktualizace hodnot** všech parametrů v celé oblasti vyžaduje cca $10^{11} \times 100 \times 96 \doteq 10^{15}$ operací.
- Sekvenční počítač s 1Gflop bude potřebovat $10^6$ sekund $\doteq 280$ hodin $= 11$ dní.
- To je ale pozdě, protože modelujeme počasí pro příští 2 dny.
- Paměťový problém (data se nevejdou do hlavní paměti sekv. počítače a musí být odkládána na disk) může řešení ještě mnohonásobně zhoršit!
- Pro výpočet spolehlivého modelu počasí je třeba mnoho iterací!!!!!

$\implies$ rozdělení dat do pamětí mnoha PE a // zpracování s pravidelnou výměnou dat je jediné schůdné řešení.

## Problémy ve světě // počítačů

- Je rozsáhlý a různorodý.
- je mladý a fluktuující (mnoho nápadů, systémů a firem vzniká a zase mizí).
- Má mnoho dimenzí a možných pohledů (paralelismus může existovat na mnoha úrovních výpočetních systémů).

## Jemnozrnný paralelismus

- Paralelismus je skryt uvnitř procesoru, který navenek provádí sekvenční nebo téměř sekvenční kód.
  - **Proudové zpracování instrukcí** a **aritmetických operací**.
  - **Superskalární** či **VLIW** ILP: Několik jednotek procesoru pracují nezávisle a paralelně.
  - **SIMD** systémy: menší jednotky provádějí synchronně // operace nad poli dat pod řízením centrálního řadiče.
  - **Vektorové procesory**: proudově pracující jednotky s dlouhými registry provádějí operace nad vektory dat.

# Hrubozrnný paralelismus

- Jednotlivé PE provádějí // procesy/vlákna.
  - **Systolická pole** a **celulární systémy**: pravidelné sítě malých PE s lokální komunikací.
  - Počítače **řízené tokem dat**, **inferenční**, **neurální**: non-von Neumannovské architektury.
  - **Symetrické** multiprocesory (s centrální sdílenou pamětí) (SMP): procesory jsou připojeny k centrální společné paměti.
  - Multiprocesory s **distribuovanou pamětí** (DMP): autonomní PE propojené propojovací sítí.
  - // **vektorové** počítače (PVP): SMP či DMP s vektorovými uzly.
  - **Svazky** (`clusters`) či `System Area Networks` (SAN): standardní PC, pracovní stanice, či SMP propojené rychlou lokální sítí.
  - **Metapočítače**: distribuované výpočetní systémy virtuálně sjednocené SW.

# Základní Flynnova taxonomie



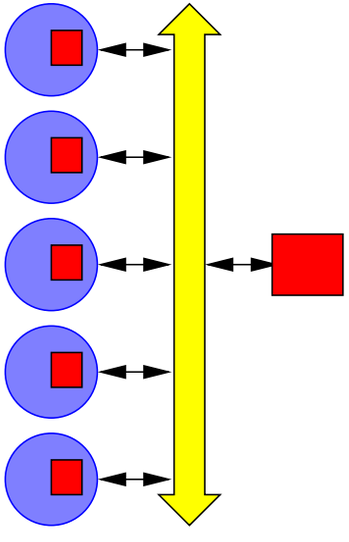
- SIMD = `Single Instruction Multiple Data`
  - Mnoho menších procesorů provádí synchronně tentýž tok instrukcí nad svými daty.
  - Samostatné SIMD počítače vymřely v polovině 90. let.
  - Principy SIMD se znovuobjevují na úrovni vektorových a multimediálních rozšíření moderních VLSI procesorů (MMX, SSE, 3DNow).
- MIMD = `Multiple Instruction Multiple Data`
  - Samostatné počítače provádějící asynchronně s občasnou synchronizací a komunikací své programy nad svými daty.
  - Prakticky všechny současné nevektorové paralelní počítače.

# SMP

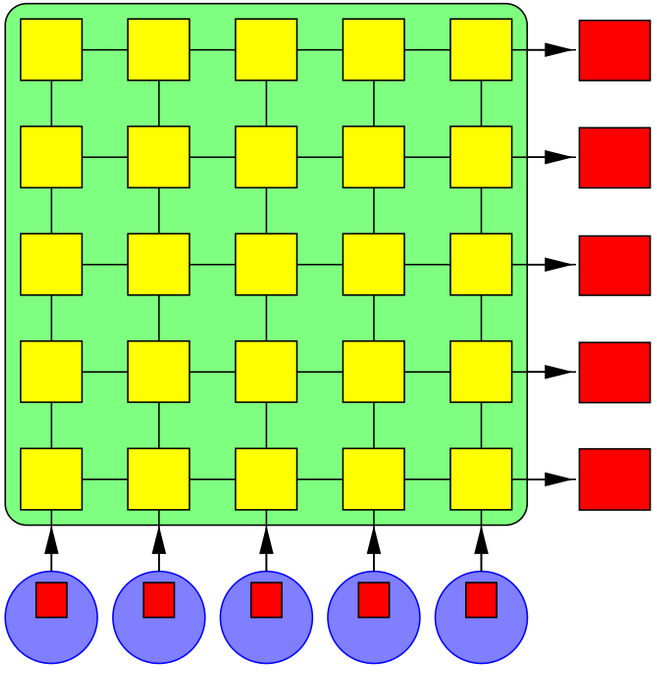


- PE = výkonné procesory s velkými skrytými pamětmi.
- Centrální sdílená paměť (1 čí více bank).
- Nutnost synchronizace přístupu k sdíleným datům.
- Škálovatelnost limitovaná propustností paměťového rozhraní:

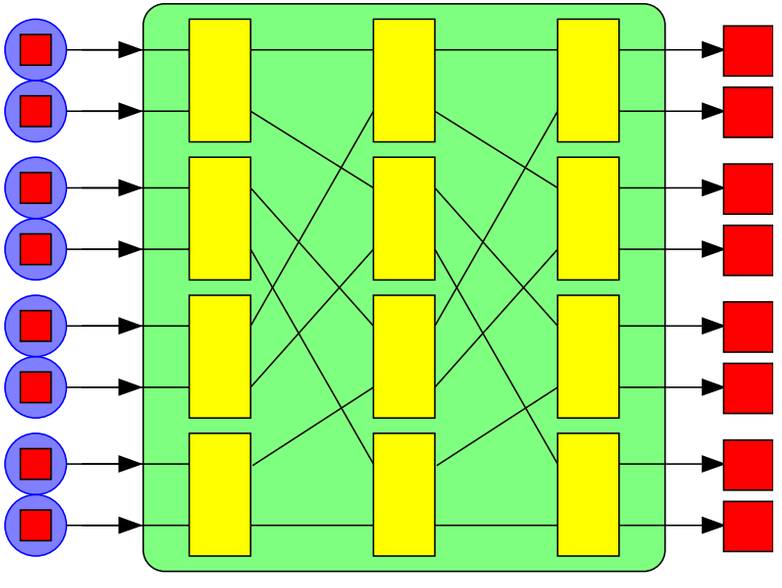
(a) **centrální sběrnice**: kolem desítky procesorů,
(b) **křížový přepínač** (`crossbar`): několik desítek procesorů (vysoká cena),
(c) **nepřímá vícestupňová síť**: levnější ale chudší náhražka kř. přepínače.



(a)



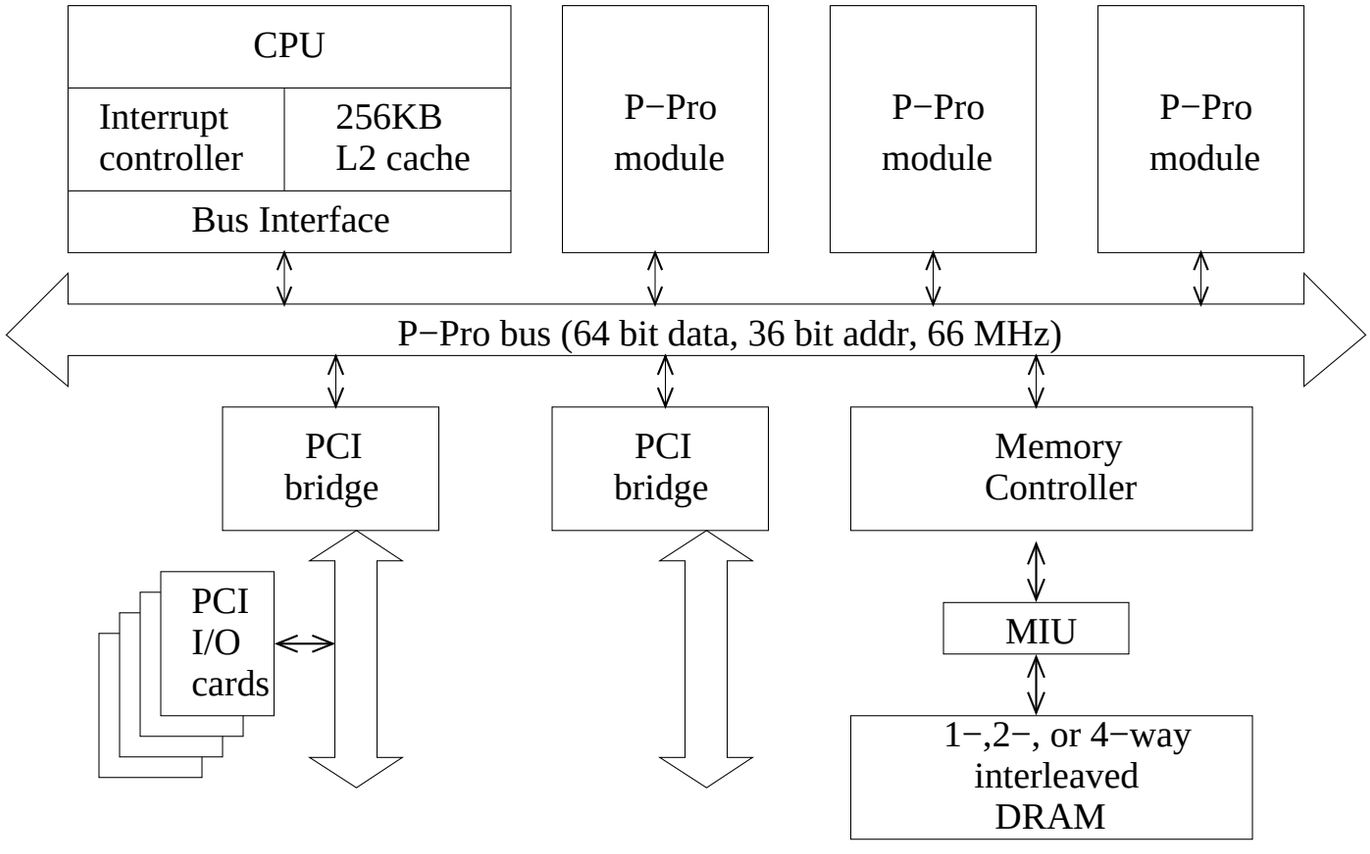
(b)



(c)

# Příklad SMP

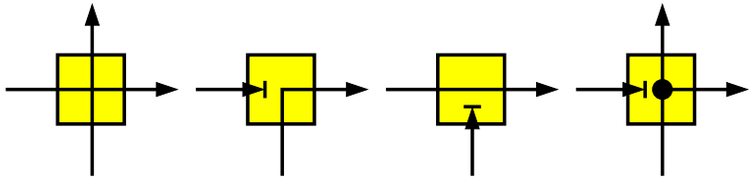




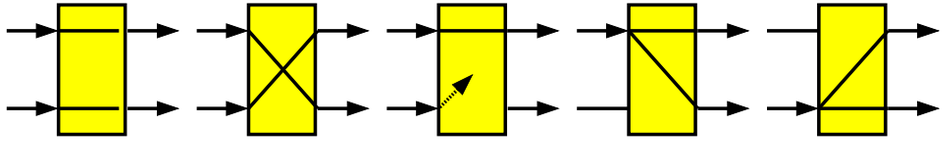

- Typická fyzická a logická organizace 4-procesorové SMP desky.
- 1 PE obsahuje na 1 čipu CPU s L1 a L2 skrytou pamětí, řadič přerušení a řadič multiprocesorové proudově pracující sběrnice (528 MB/s), která je připojena
  - přes řadič paměti a `Memory Interleave Unit` (MIU) na DRAM rozdělenou do 4 bank,
  - přes mosty na 2 nezávislé PCI sběrnice (displej, síť, SCSI, V-V kanály).

# Implementace přepínačů

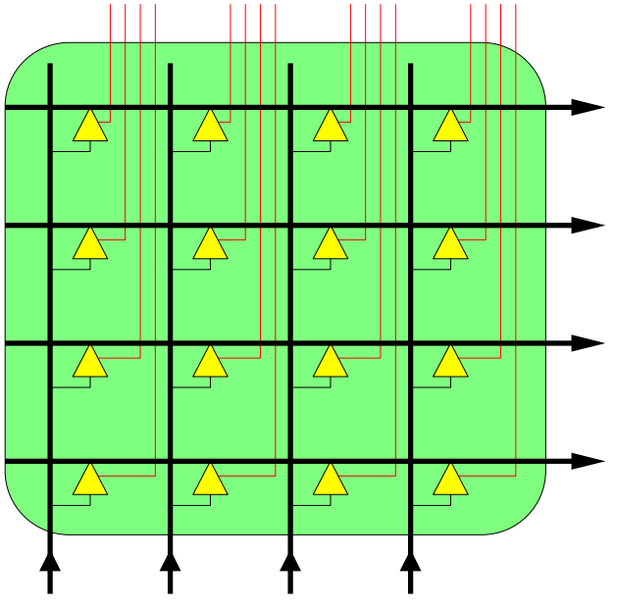




(a)



(b)



(c)

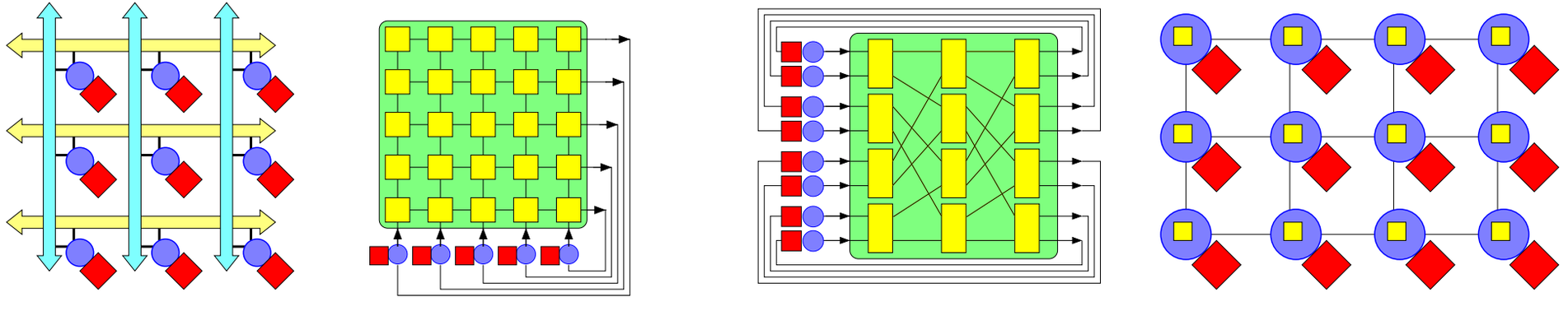
(a) Přepínací prvek křížového přepínače.

(b) $2 \times 2$ přepínací prvek nepřímé sítě (elementární křížový přepínač).

(c) Implementace křížového přepínače (jedna možná).

- PE = výkonné samostatné počítače s lokálními (soukromými) pamětmi.
- Ale mohou to být taktéž SMP!!!!!
- Všechny procesory mohou současně přistupovat do svých lokálních pamětí.
- V/V jsou distribuované a škálovatelné podobně jako paměť.
- Škálovatelnost je mnohem vyšší, neb je limitovaná celkovou propustností propojovací sítě, a ta je závislá na rychlosti kanálů a vlastnostech topologie (stupeň souvislosti, bisekční šířka, atd.):

(a) 2-D **mřížka sběrnic**.
(b) **křížový přepínač**
(c) **nepřímá vícestupňová síť**
(d) **přímá síť**

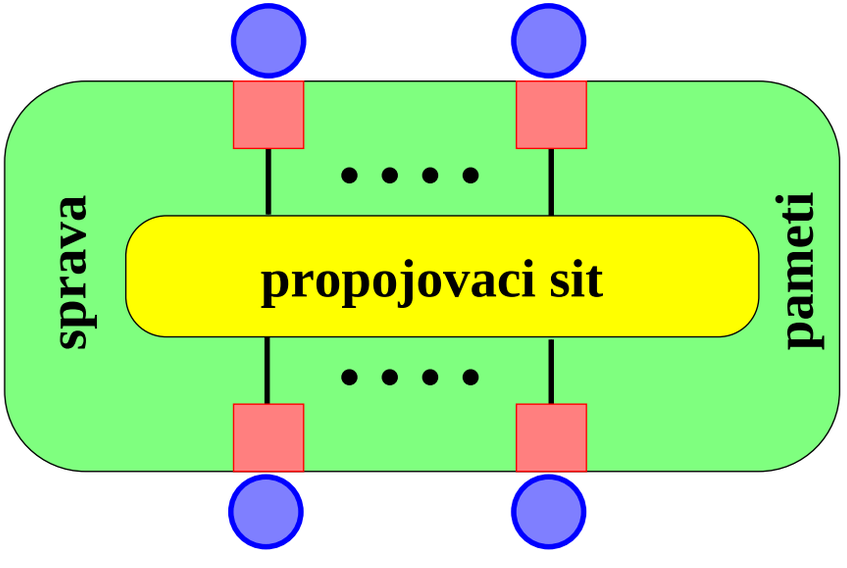


(a) (b) (c) (d)

# Paměťové modely DMP

## Sdílený (společný) adresní prostor (SAP)

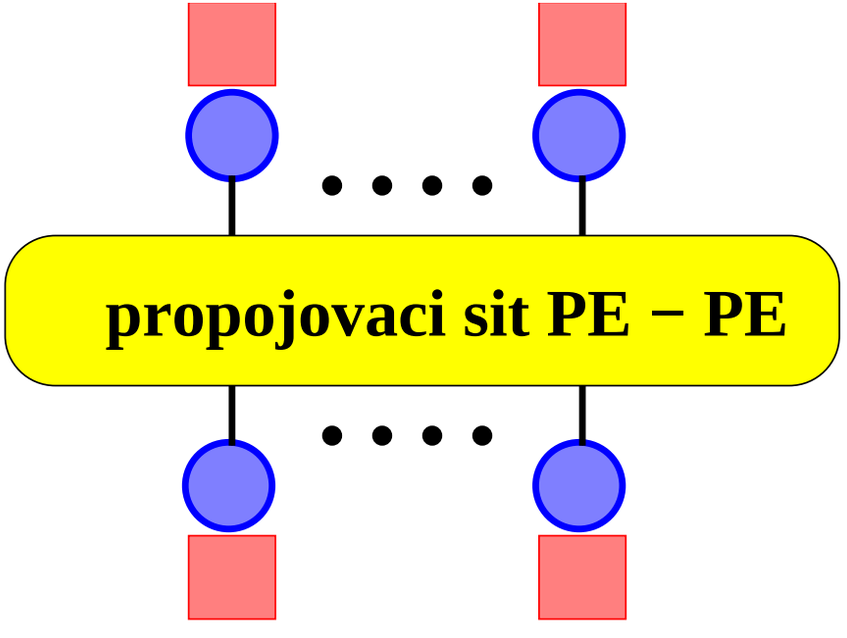
- Fyzicky oddělené paměti jsou adresované jako globální společný AP (jako v SMP).
- Jakýkoli PE se může odkázat přímo na jakoukoli adresu AP.
- 2 stejné adresy na 2 různých PE odkazují na **totéž** paměťové místo.
- Meziprocesorová komunikace je **implicitně** skryta v operacích Read-Write (Load-store).
- Jazyková podpora: OpenMP standard (`www.openmp.org`).



# Distribuovaný (oddělený) adresní prostor (DAP)



- Každý PE má privátní adresní prostor, do kterého nemají ostatní PE přímý přístup.
- Jakýkoli PE se může odkázat přímo na jakoukoli adresu AP.
- 2 stejné adresy na 2 různých PE odkazují na **různá** paměťová místa.
- Meziprocesorová komunikace je založena na **explicitním zasílání zpráv**.
- Zasílání zpráv má sémantiku **volání vzdálených funkcí** (`Remote Procedure Call` (RPC)).



## Vlastnosti a výhody a nevýhody modelu SAP

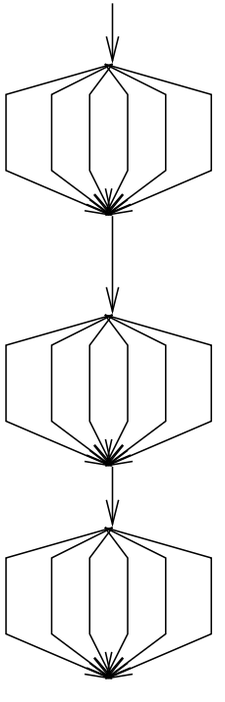
- SAP je jednodušší a snazší pro programování, zvláště pokud jsou komunikační vzory složité a dynamicky proměnné.
- Pro SAP je jednodušší i kompilace // programů.
- Nutná HW podpora (čistě SW implementace jsou pomalé a neefektivní): každý `Load/Store` se musí analyzovat a případně přeložit do operací meziprocesorové komunikace.
- Pro uživatele je obtížné optimalizovat komunikační náklady.
- SAP na SMP = architektura `Uniform Memory Access (UMA)`: všechny procesory to mají do dané paměťové buňky stejně daleko.
- SAP na DMP = architektura `NonUniform Memory Access (NUMA)`: přístup do lokální buňky je rychlejší než do vzdálené.
- Alternativní název pro SAP na DMP je `Distributed Shared Memory (DSM)`.
- Pokud je zajištěna koherence skrytých pamětí, pak se jedná o architekturu `Cache-coherent (CC)` NUMA (viz příští přednáška).
- Efektivní zajištění koherence skrytých pamětí není snadné a limituje škálovatelnost.

## Vlastnosti a výhody a nevýhody modelu DAP

- Jednodušší HW.
- Lepší škálovatelnost.
- Komunikační operace musí být implementovány v paralelním programovém prostředí, typicky jako komunikační knihovna. Nejvýznamnější příklad je MPI.
- Programátor je nucen se soustředit na komunikační strukturu programu a optimalizovat ji, neboť
- komunikace je explicitní a snadněji pochopitelná.
- Synchronizace se přirozeně spojena s komunikací.
- Lze snadno implementovat (simulovat) i na SMP.

## Škálovatelnost a omezený stupeň paralelismu v daném algoritmu

**Příklad 1:**
Uvažujme algoritmus, který se skládá ze sekvenční části (musí provádět pouze 1 procesor) a paralelizovatelné části (lze provést $p$ procesory paralelně) (viz obr., typický program v OpenMP). Uvažujme **normalizovaný sekvenční** čas $1 = f_s + f_p$, kde $f_s$ a $f_p$ jsou podíly sekvenční a paralelizovatelné části. Jak může být velká sekvenční část, pokud chceme na $p = 100$ procesorech dosáhnout zrychlení $z = 80$?



**Řešení:**
Dle definice, zrychlení je poměr sekvenčního a paralelního času.

$$z = \frac{1}{f_s + \frac{f_p}{p}} = \frac{p}{(p-1)f_s + 1}.$$

Odsud dostaneme

$$f_s = \frac{p - s}{s(p-1)}.$$

Čili v našem případě, $f_s \doteq 0.25\%$. ♣

**Příklad 2:**
Uvažujme SAP DMP s $p = 32$ procesory pracujícími na $f = 1$GHz. Předpokládejme pro jednoduchost, že všechny paměťové operace se trefí do lokální vnitřní paměti a nedochází k výpadkům a že každý procesor dosahuje v případě lokálního výpočtu základního IPC $IPC_{\text{base}} = 2$. Přístup do vzdálené paměti (Load nebo Store 1 buňky paměti) jiného procesoru trvá čas $t_m = 400$ns. Kolikrát se zpomalí výpočet, ve kterém bude $RA_{\text{rate}} = 0.2\%$ odkazů do vzdálené paměti v porovnání s výpočtem bez jakékoli komunikace (pouze lokální `Load/Store`)?

**Řešení:** Procesor při vzdáleném přístupu musí čekat $RA_{\text{cost}} = \frac{400\text{ns}}{1\text{ns}} = 400$ cyklů a o to se mu zvýší průměrný počet cyklů na instrukci

$$CPI_{\text{new}} = CPI_{\text{base}} + RA_{\text{rate}} \times RA_{\text{cost}} = \frac{1}{IPC_{\text{base}}} + RA_{\text{rate}} \times RA_{\text{cost}} = 0.5 + 0.002 \times 400 = 1.3.$$

Čili výpočet se zpomalí $\frac{CPI_{\text{new}}}{CPI_{\text{base}}} = \frac{1.3}{0.5} = 2.6$ krát. ♣

## Zdroje informací na Internetu

`http://www.top500.org/`

`http://www.hpcwire.com/`

# Počítačové svazky (clusters)

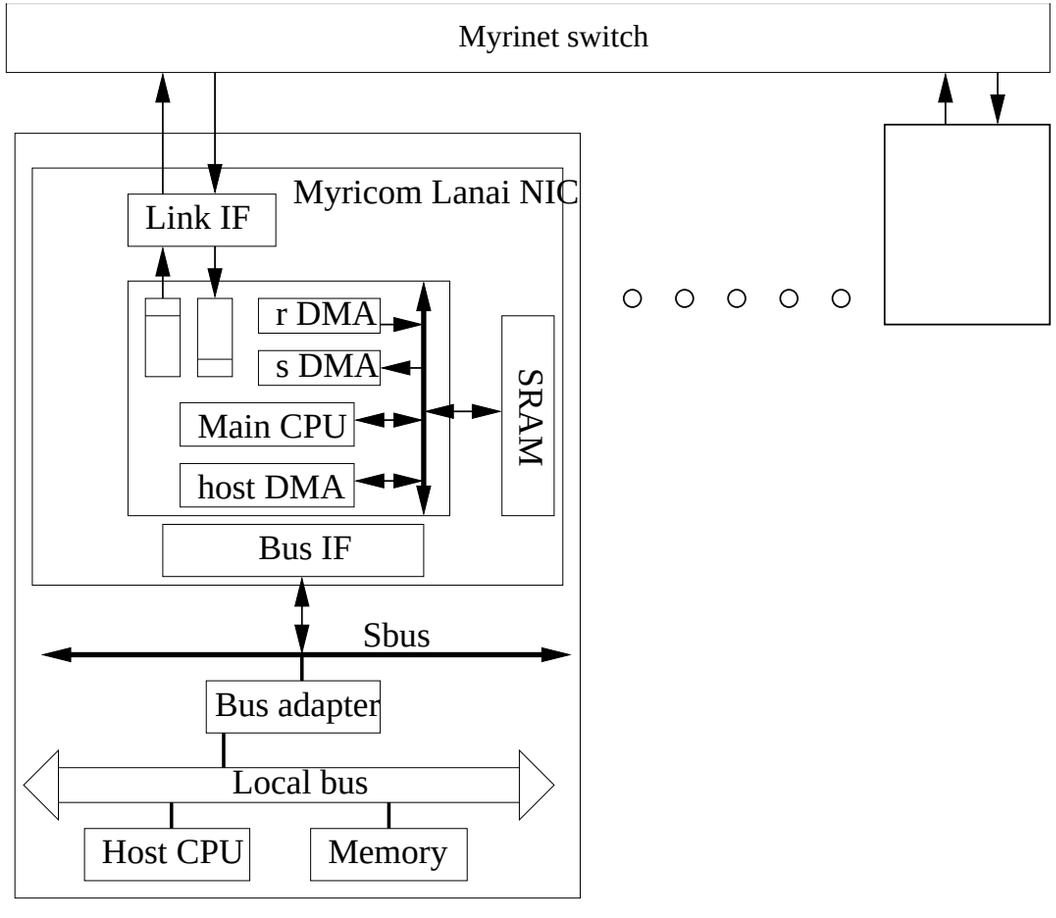


- Výrazný nástup od roku 1994 (NASA zprovoznila tzv. Beowulf cluster).
- Od té doby se svazky vyvinuly do stavu plně srovnatelného se speciálními // architekturami.
- Výhody: nízká cena HW i SW (seriově vyráběné komponenty (PC) a většinou Open Source Software).
- Téměř výhradně Linux.
- Dnes dominují v seznamu Top500.
- Komerční podpora, většina velkých výrobců a mnoho nových firem se na svazky zaměřuje, dodávky na klíč (konfigurátoři), např. `www.linuxnetworx.com`.
- IEEE Task Force on Cluster Computing, mnoho konferencí.
- Klíčový význam má pro svazky propojovací síť.
  - Všechny mají 100 Mb/s Ethernet nebo Gigabit Ethernet: nízká cena, ale velká latence (100 $\mu$s).
  - Dnes nejlepší řešení jsou System Area Networks (SAN): Myrinet (200 MB/s), InfiniBand (850 MB/s), SCI (500 MB/s). Latence řádově jednotky $\mu$s.
  - Všechny založeny na menších přepínačích $(8, 16, 32)$, které se zapojují do nepravidelných či pravidelných topologií (např. vícestupňové Closovy sítě).

# Srovnávací tabulka SAN z r. 2003

| Síť | InfiniBand | Myrinet | Gigabit Ethernet |
|---|---|---|---|
| Latence přepínače | 160 ns | 200 ns | 10,000 ns |
| End-to-End latence (8B paket) | 7.6 $\mu$s | 8 $\mu$s | 60 $\mu$s |
| End-to-End latence (1KB paket) | 13 $\mu$s | 22 $\mu$s | 80 $\mu$s |
| Průtočnost | 822 MB/sec | 250 MB/sec | 100 MB/sec |
| Režie CPU | 3% | 6% | 80% |

# Architektura síťové Myrinet karty



# Příklad realizace sítě Myrinet/InfiniBand



- Síť Myrinet pro 256 PE tvořená 80 křížovými přepínači $8 \times 8$.
- PE budou připojeny k listovým přepínačům.