13. Architektury paralelních počítačů

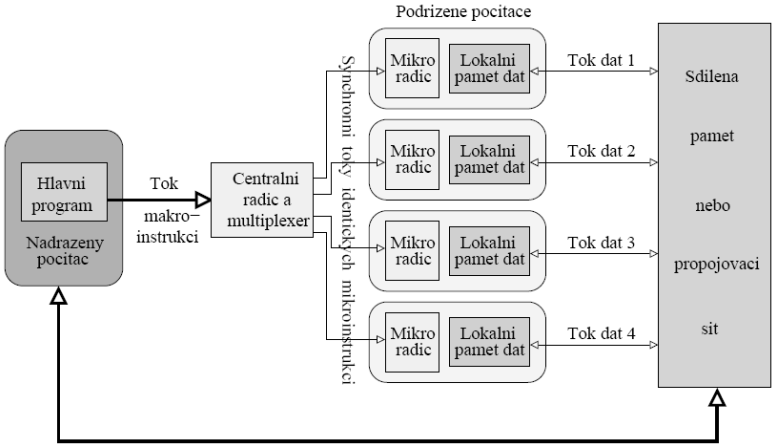
Princip činnosti

Systém, v němž probíhá několik procesů současně. Snaha zvyšovat výkonnost, zvýšení bezpečnosti a spolehlivosti. Zcela přirozená vlastnost numerických i ostatních algoritmů. (pro analogové počítače naprostá samozřejmost)

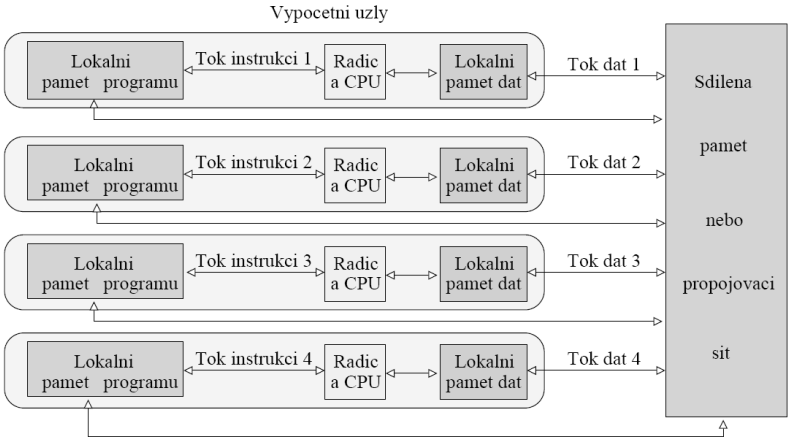
Úrovně granularity:
1)Příkazy instrukce
2)Cykly, iterace
3)Podprogramy
4)části úloh a programů
5)nezávislé úlohy a programy

Taxonomie paralelních systémů, Flynnova klasifikace.

Velký počet vedl o pokus zavedení taxonomie (klasifikace).
Flynnova klasifikace lze par. sys. třídit z hlediska počtu toků instrukcí a dat. Systémy s 1 tokem instrukcí SI (Single instruction stream), systém s několika toky instrukcí MI (Multiple Instruction stream) Analogicky SD (Single Data) MD (Multiple Data)



Ilustrace 1: SIMD

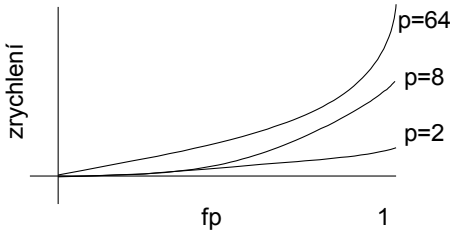


Ilustrace 2: MIMD

SISD počítač, zpracovávající data sériově podle 1 programu. typicky von Neumannův typ.

SIMD počítač používající větší množství stejných jednotek řízených společným programem Přitom data zpracovávaná v jednotlivých procesorech jsou různá. Každý jinou hodnotu podle sejné instrukce. Takovéto počítače se dnes již nevyrábí. Výpočet musí být synchronní. Větší počet jednodušších procesorů.

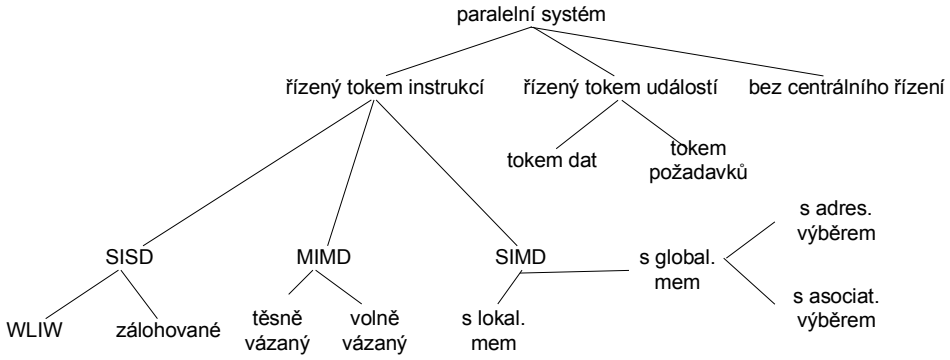
MIMD je multiprocesorový systém, v němž každý procesor je řízen samostatným programem a pracuje s jinými daty než ostatní procesory. STAR na fakultě je MIMD. Lze si představit jako paralelně propojené samostatné počítače. Teoreticky může každý provádět jiný program. Výpočet může být asynchronní nebo synchronní (použití bariér).



MISD kategorie, která vznikla uměle na základě tyto klasifikace a není v praxi běžná.

MSIMD systém v němž pracuje několik SIMDů

SPMD (Same program Multipe Data stream) modifikace SIMD, všechny procesory sice provádějí stejný program ale nezávisle na sobě (bez synchronizace)



- SIMDy i MIMDy umí reprezentovat jak sdílenou tak distribuovanou paměť v závislosti na tom, zda používají sdílenou paměť nebo propojovací síť.
- Provedení příkazu `if` nebo `case` v SIMD:
  - každý procesor provede vyhodnocení podmínky
  - nastává časový multiplex = procesory, které vyhodnotily podmínku jako `true` pokračují ve výpočtu
  - ve druhém kroku se multiplex přepne a stojí zase ty, které pracovali
  - nesmí se totiž stát, že v SIMD si dělí 2 procesory různé věci!
- SIMD a MIMD jsou asymptoticky výpočetně ekvivalentní
  - simulace SIMD pomocí MIMD je triviální
  - simulujeme-li MIMD pomocí SIMD, pracuje se na úrovni časových multiplexů jako při provádění `if` a `case`. Výpočet bude na SIMD trvat tolikrát déle, kolik různých instrukcí MIMD provádí.

Klasifikace podle organizace paměti

Sdílená paměť' – „komunistická“ architektura. 2 procesory komunikují pomocí zápisu (čtení) do (ze) sdílené paměti. (UMA – Uniform Memory Access)
Distribuovaná paměť' - „kapitalistická“ architektura (Tvrdíkovina, to jsem si nevymyslel já :)). 2 procesory komunikují pomocí zpráv. Nelze přistupovat do něčí paměti bez dovolení. (NUMA – Non UMA = rychlost přístupu do vlastní paměti je jiná než do cizí paměti). Umožňuje libovolnou škálovatelnost.
Virtuálně sdílená paměť' – ve reálu se jedná o distr. paměť, ale řešení přístupu do ní je řešeno hardwarově (typicky). Nemusí tedy každý přístup do paměti řešit procesor (majitel paměti) a každý může s libovolnou pamětí pracovat jako se sdílenou pamětí. Na pozadí je tedy vlastně zápis/čtení převedeno na zprávy. (CC-

NUMA Cache Coherent – NUMA)

Zajištění konzistence dat ve VP a VP a HP je přímá signalizace mezi VP a VP a HP. ***Write Through***t = zapiš ihned, když změníš data, ***Write back*** = zapiš do paměti až při uvolňování bloku a to jen v případě, že jsi inkriminovaný blok dat během výpočtů změnil (jinak to je zbytečné).

Cache Coherent Protokoly – k zajištění konzistence dat v paměti
Protokol Write Through Write Non Allocate – ***WTWNA***

1) Procesor 1 nenajde data v cache a proto musí provést čtení z paměti.
2) Stav bloku v cache u P1 je nyní valid (platný).
3) Procesor 3 také nenajde data v cache a proto musí provést čtení z paměti.
4) Nyní mohou oba procesory opakovaně nezávisle číst položku X.
5) Procesor 3 změní hodnotu X v cache a také v hlavní paměti
6) Řadič cache u P1 odposlechne zápis na sběrnici a zneplatní blok X.
7) (nový stav bloku X v cache u P1 je invalid což přinutí P1 přečíst novou hodnotu.)

Write Back invalidation protocol – ***MESI*** (modified, exclusive, shared, invalid)

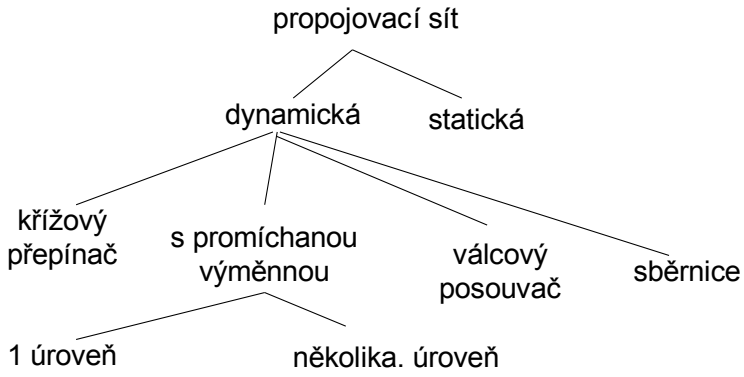
1) Procesor 1 nenajde data v cache a proto musí provést čtení z paměti.
2) Kontrolér (KSP) u proc.2 a proc.3 indikují signálem S=0, že nemají kopii bloku X proto je stav bloku u proc.1 Exclusive.
3) Při čtení X z paměti indikuje KSP u procesoru 1, že daný blok již má ve své
4) cache nastavením S=1. To způsobí, že nový stav bloku jak u proc.3 tak u proc.1 je Shared.
5) Procesor 3 zjistí, že chce zapisovat do bloku ve stavu Shared, proto umístí na sběrnici transakci `BusUpgrd` kterou zneplatní všechny ostatní kopie daného bloku v ostatních cache. `BusUpgrd` je krátká a nenese data, paměť není modifikována.
6) Procesor P1 při načítání obdrží aktuální hodnotu X přímo z cache procesoru 3. Při této příležitosti je též modifikována hodnota v hlavním paměti. V obou cache přechází blok do stavu Shared
7) Procesor 2 chce změnit hodnotu X, kterou nemá v cache, proto vygeneruje transakci `BusRdX`, která načte daný blok a zároveň zneplatní všechny ostatní kopie. Nový stav bloku v cache u proc.2 je Modified.

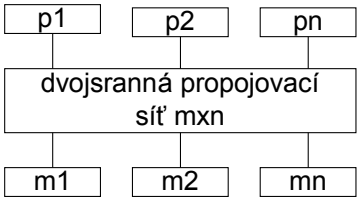
Těsně vázaný MIMD - procesor nemá vlastní paměť, nebo je velmi malá. malý stupeň autonomie.

Volně vázané MIMD - každý procesor vlastní lokální HP a vlastní PZ, značný stupeň autonomie, HP pojme data i program,rys- jeden OS a řeší jedenu úlohu. Komunikace formou zpráv. PA (překladač adres) schopnost řešit konflikt při přístupu do propojovací sítě. 3ířka toku dat Malá 1 bit. nevyžaduje silnou interakci.

Propojovací sítě, statické / dynamické

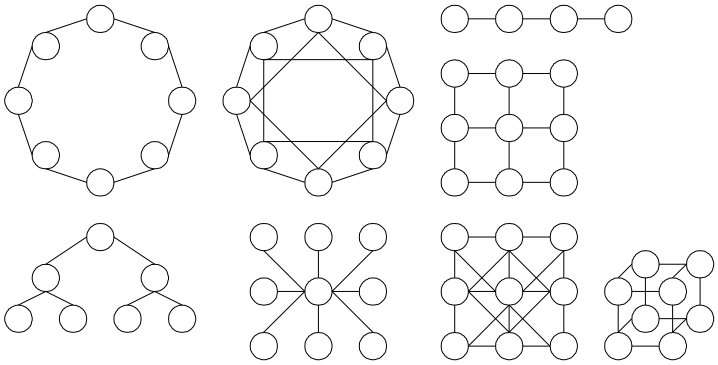
Problém komunikace mezi jednotkami. jednostranná složitější, každý s každým. U dvojstranné vyžadujeme pouze spoje p do m. neumožňuje spoje mezi. p x p. je to v podstatě přepínač. Důležitá i strategie řízení.





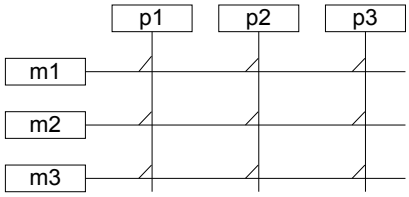
Statické – v nichž spojovací cesty zůstávají neměnné.

Typ	průměr	stupeň
Cesta	N-1	2
Kružnice	N/2	2
2xkruž	N/4	4
bin.strom	2[log(N+1)-1]	3
Krychle	logN	logN

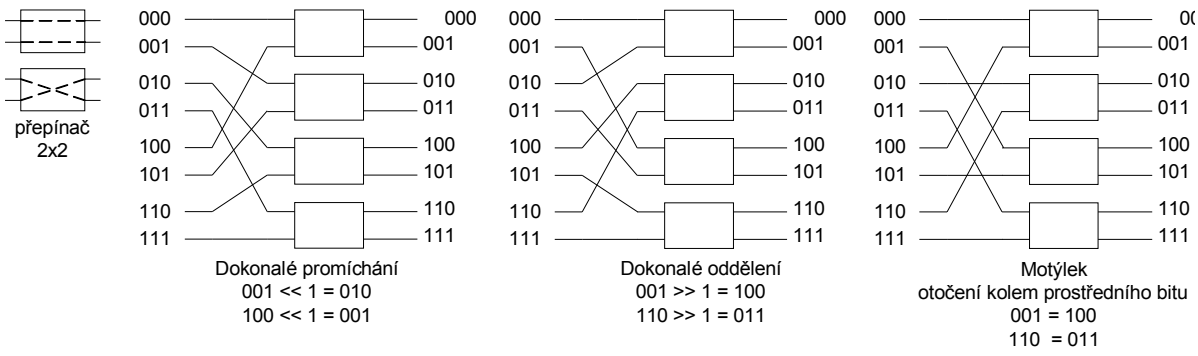


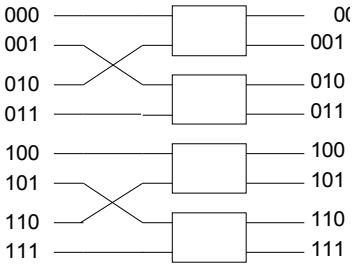
Dynamické - spoje volně vznikají a zanikají (prvky které umožňují spojovat).

- ***křížový přepínač nebo jen trojúhelníkový***
  - hlavní výhodou je vzájemná nezávislost propojovacích cest, která umožňuje současně realizovat tolik spojení, kolik dvojic In/Out lze vytvořit.
  - vysoká cena.



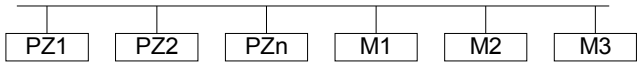
- ***propojovací sítě typu promíchání s výměnou***
  - nejvíce se osvědčily přepínače sestavené z elementárních přepínačů a x b
  - dělí se na jednoúrovňové a víceúrovňové.





1/2 Motýlek (můj název)
otočení kolem prostředního bitu
001 = 100
110 = 011

- ***Sběrnice*** - nejjednodušší prostředek, snadno se přetíží



Víceúrovňové
- ***blokující*** - spojení jednoho vstupu na výstup, které je dosud volné, může být blokováno již existujícím spojením.
  - Omega - 3x dokonalé promíchání
  - Základní síť = Dokonalé oddělení + ½Motýlek
  - SW-banyan = ½ motýlek + motýlek
- ***přestavitelné*** - spojení lze dosáhnout za cenu přestavění již existujícího spoje (Benešova síť = dokonalé oddělení + ½ motýlek + ½ motýlek + dokonalé oddělení)
- ***neblokující*** - nejdokonalejší. lze vytvořit libovolné spojení, bez ohledu na již uskutečněná (Closova)

Růst rychlosti, Amdahlův zákon, oblasti využití par. systémů

Problém růstu výkonnosti systému jako celku
- ztráty spojené s komunikací
- nedokonalé vytížení procesorů
- neznalost vhodných algoritmů
- zrychlení (lineární (ideálně), superlineární(pouze v některých případech), zpomalení – často :))

Amdahalův zákon - čistě paralelních úloh velmi málo, většinou kombinováno se sériovým zpracováním.
- S_Z součinitel zrychlení (poměr doby, která by byla potřebná, kdyby celý výpočet probíhal na 1 procesoru k době potřebné při částečné paralelizaci)
- f_S podíl sériové délky
- f_P podíl paralelní délky (fs + fp = 1)
- t celková doba sériového výpočtu

použijeme-li p procesorů klesne délka paralelizované části na $\frac{t \cdot f_P}{p}$

$$S_Z = \frac{t}{t \cdot f_S + \frac{t \cdot f_P}{p}} = \frac{1}{(1 - f_P) + \frac{f_P}{p}}$$

Oblasti použití

- modelování a simulace, přírodní vědy
- automatizace inženýrských prací (návrh VLSI, WSI, dynamika, aerodynamika)
- řešení úloh umělé inteligence, vojenská technika
- výzkum energetických zdrojů
- zpracování signálů