Paralelní programování

přednáška 5

Michal Krupka

15. března 2011

Ještě ke kritickým sekcím

Použití v praxi

- obvykle pomocí **zámků** (**locks**)
- klasické objekty (někdy libovolné, někdy speciální třídy)
- mají metody na vstupní a výstupní protokol (enter, exit)
- kritická sekce se označí pomocí klíčového slova

Příklad v C#

```
Object thisLock = new Object();
lock (thisLock)
{
  // Kritická sekce
}
```

Cvičení

1. V globálním poli accounts jsou uloženy zůstatky bankovních účtů. Napište funkci transfer, která převede zadanou částku mezi účty: (transfer a1 a2 amount). Můžete použít pomocná globální data.

Ještě ke kritickým sekcím

Cvičení

2. Vylepšete funkci transfer tak, aby šlo současně (bez čekání) převádět částku mezi různými dvojicemi účtů.
3. Vylepšete funkci transfer tak, aby kromě požadavků z předchozího příkladu umožňovala provádět účetní uzávěrku, tj. test, zda součet zůstatků na všech účtech odpovídá očekávané hodnotě (jelikož provádíme pouze přesuny mezi našimi účty, musí být tento součet konstantní). Udělejte to tak, že napíšete funkci balance, která vrátí součet zůstatků na všech účtech. Tato funkce musí za všech okolností vracet stejnou hodnotu. Může dočasně zablokovat převody mezi účty (tj. způsobit čekání ve funkci transfer). Pozor na uváznutí!

Flynnova taxonomie

- klasifikace architektur počítačů podle možnosti
  - paralelního vykonávání instrukcí
  - paralelního zpracování dat
- Michael J. Flynn, 1966

	Single Instruction	**Multiple Instruction**
Single Data	SISD	MISD
Multiple Data	SIMD	MIMD

SISD

Klasická von Neumannova architektura (PC s jedním procesorem)

MISD

Neobvyklé, např. kvůli zabránění chybám (raketoplán)

SIMD

Paralelní zpracování dat stejným algoritmem (některé superpočítače, vektorové procesory, GPU)

MIMD

Více procesů zpracovává různá data (moderní PC, distribuované systémy)

Flynnova taxonomie

SISD

- jeden procesor provádí jeden proud instrukcí, pracuje s daty uloženými v jedné paměti

Příklad: klasická von Neumannova architektura

MISD

- více procesorů provádí současně více programů s týmiž daty

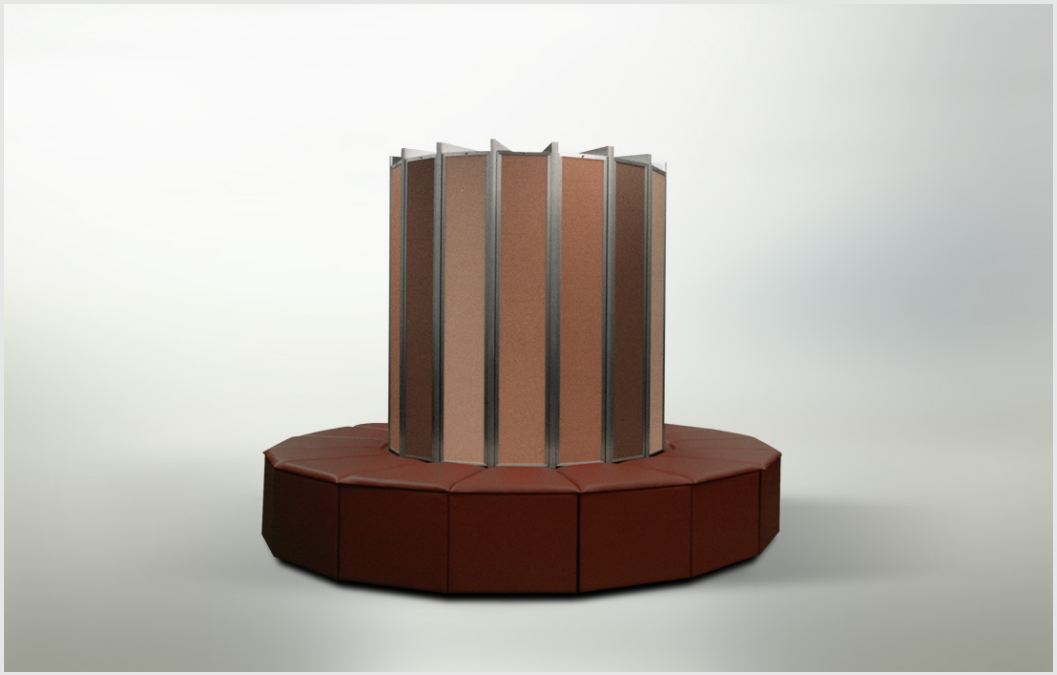
Příklady: nepříliš obvyklé, používá se kvůli předcházení chybám. Např. raketoplán (pět počítačů zpracovává stejná data).

Flynnova taxonomie

SIMD

- více procesorů provádí současně (ve stejném taktu) jeden proud instrukcí na různých datech
- mezi procesy není nutná synchronizace
- masivně paralelní výpočty

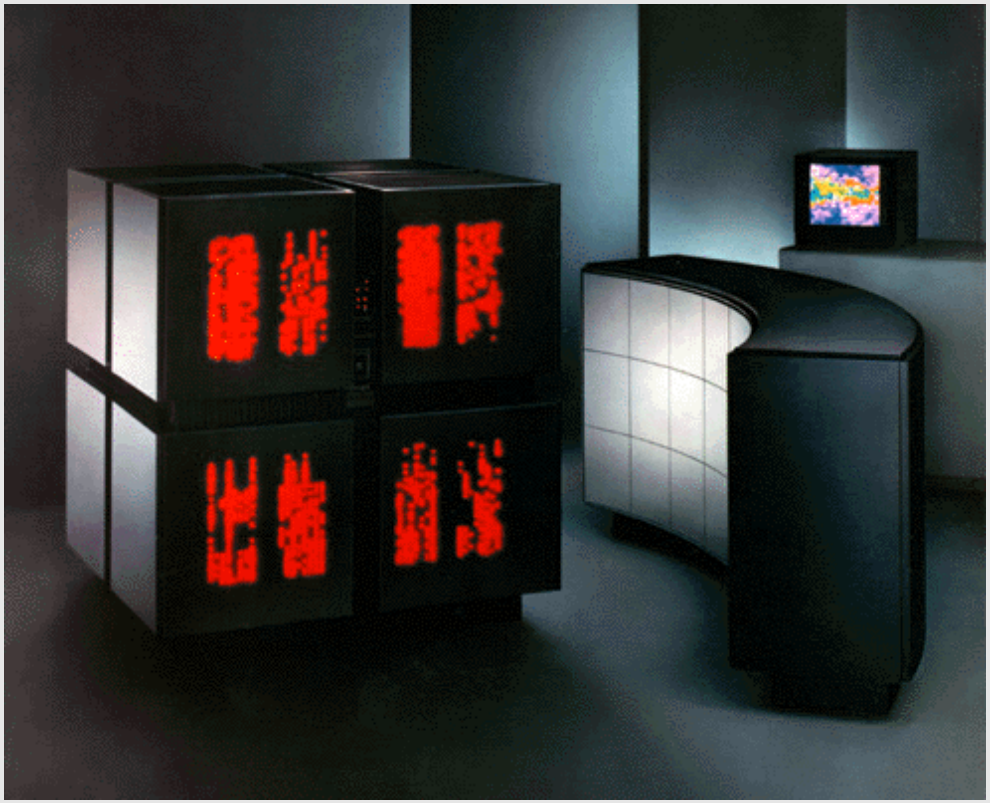
Příklady: Superpočítače v 70. a 80. letech, např. **Cray:**



- Cray-1, 1976
- 200 000 hradel
- 1 662 „procesorů“ 113 variant
- vektorové instrukce
- pohovka zdarma

Flynnova taxonomie

Connection Machine:



- CM-1, 2, konec 80. let
- 65 536 jednobitových procesorů,
- uspořádaných do 12-rozměrné hyperkrychle
- umělá inteligence

Další:

- vektorové instrukce v procesorech (UltraSPARC I: VIS, x86: MMX, POWER: AltiVec)

Flynnova taxonomie

MIMD

- více procesorů provádí současně více úloh na různých datech
- mezi procesy je nutná synchronizace

Příklad: moderní osobní počítače i superpočítače, distribuované systémy

Podskupinou je **SPMD: Single Program/Multiple Data**

- jednotlivé procesory vykonávají tentýž program (s různými daty), ale ne nutně ve stejném kroku,
- je nutná synchronizace pomocí **bariér** (viz dále)

Příklady

- nejběžnější způsob paralelního programování i na MIMD,
- moderní GPU: stovky jader, souběžně **tisíce procesů**
  - na některé úkoly mnohonásobně rychlejší než CPU
  - programové modely: CUDA (NVIDIA), OpenCL (Apple, otevřený standard, široce podporovaný)
  - aplikace: zpracování obrazu, modelování a simulace, výpočetní chemie, biologie, medicína...

Bariéry (Jordan 1978)

Synchronizační nástroj, používaný zejména při SPMD programování.

Bariéra

Místo v programu, na kterém procesy čekají, dokud jej všechny nedosáhnou.

Příklad: výpočet n-tého řádku Pascalova trojúhelníku pomocí n procesů.

0	1	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0
0	1	2	1	0	0	0	0	0	0	0
0	1	3	3	1	0	0	0	0	0	0
0	1	4	6	4	1	0	0	0	0	0
0	1	5	10	10	5	1	0	0	0	0
0	1	6	15	20	15	6	1	0	0	0
0	1	7	21	35	35	21	7	1	0	0
0	1	8	28	56	70	56	28	8	1	0

Obrázek: Hodnoty pole A v jednotlivých krocích algoritmu

Bariéry

Algoritmus pro výpočet n-tého řádku Pascalova trojúhelníka

N-tý řádek Pascalova trojúhelníka	
integer A[N + 2] $\leftarrow$ 0, A[1] $\leftarrow$ 1	
i-tý proces pro i od 1 do N $-$ 1	
	integer sum
1:	loop N $-$ 1 times
2:	sum $\leftarrow$ A[i $-$ 1] + A[i]
3:	barrier
4:	A[i] $\leftarrow$ sum
5:	barrier

Obrázek: N-tý řádek Pascalova trojúhelníka

Bariéry

Implementace

Ukážeme jednu možnost implementace bariéry. Nejprve bariéra pro dva procesy, implementovaná symetricky:

Bariéra pro dva procesy	
Boolean arrive[poč. procesů] $\leftarrow$ [false, . . . , false]	
Proces i	Proces j
1: await not arrive[i]	1: await not arrive[j]
2: arrive[i] = true	2: arrive[j] = true
3: await arrive[j]	3: await arrive[i]
4: arrive[j] = false	4: arrive[i] = false

Obrázek: Symetrická bariéra pro dva procesy

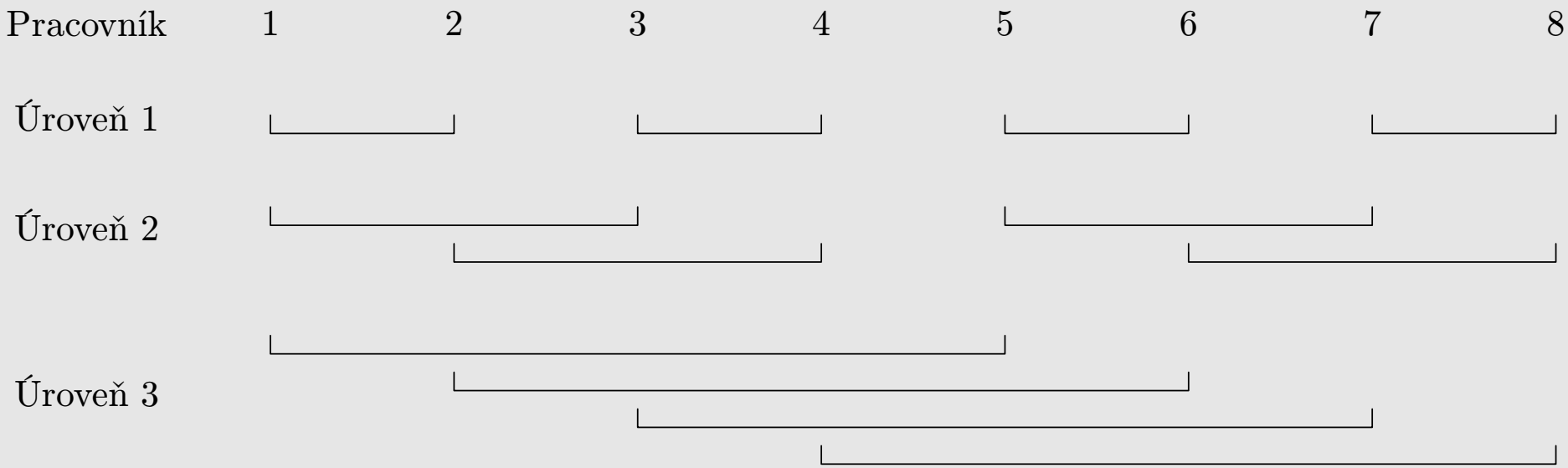
Cvičení

3. Stanovte podmínky bezpečnosti a živosti pro tento algoritmus a zdůvodněte jejich splnění.
4. Implementujte tento algoritmus.

Bariéry

Implementace pro více procesů: Butterfly (motýlkovitá) bariéra

- pro počet procesů $N = 2^X$, logaritmická složitost
- na úrovni s se synchronizují procesy ve vzdálenosti 2^s
- po projití všech úrovní se proces přímo nebo nepřímo synchronizoval s každým jiným procesem



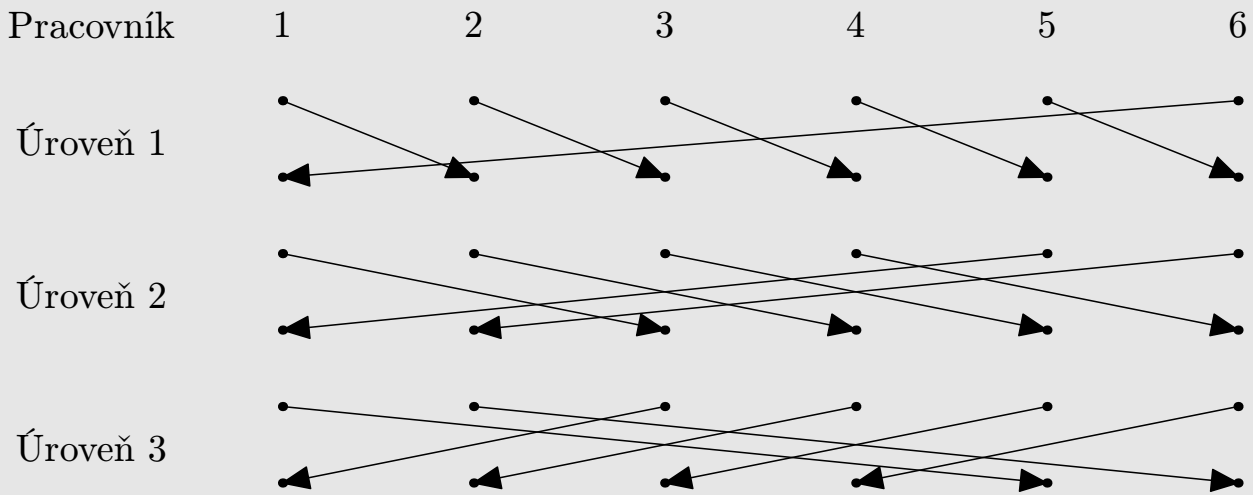
Cvičení

5. Implementujte motýlkovitou bariéru.

Bariéry

Implementace pro více procesů: Diseminační bariéra

- lepší pro počet procesů $N \neq 2^X$, logaritmická složitost
- na úrovni s se proces i synchronizuje s procesem ve vzdálenosti 2^s modulo N:
  - nastaví vlajku procesu $(i + 2^s)\ mod\ N$
  - čeká na a resetuje vlajku procesu $(i - 2^s)\ mod\ N$



Cvičení

6. Implementujte diseminační bariéru.