Paralelní programování

přednášky

Jan Outrata

únor–duben 2011

Literatura

- Ben-Ari M.: *Principles of concurrent and distributed programming*. Addison-Wesley, 2006. ISBN 9780321312839
- Andrews G. R.: *Concurrent programming: principles and practice*. Addison-Wesley, 1991. ISBN 0805300864
- Andrews G. R.: *Foundations of Multithreaded, Parallel, and Distributed Programming*. Addison-Wesley, 2000. ISBN 0201357526
- Schneider F. B.: *On concurrent programming*. Springer, 1997. ISBN 0387949429
- Magee J., Kramer J.: *Concurrency, State Models and Java Programs*. John Wiley and Sons Ltd, 1999.

Úvod do paralelního a distribuovaného programování

- dnešní programy ve své podstatě paralelní nebo distribuované: např. událostmi řízená UI, operační a řídící systémy, víceuživatelské síťové aplikace, ...
- podporováno moderními programovacími jazyky, např. Java, C#
- technologie se mění, ale stále stejné
  - **principy**: prokládání, vzájemné vyloučení, bezpečnost, živost
  - **základní problémy**: kritická sekce, product-konzument, čtenáři a písaři aj.
  - klasické **nástroje řešení**: semafor, monitor, zprávy aj. (vznikají i nové, např. dispatch queues)
- *Co je tedy nové?* Dnes je běžné, ba přímo nutné, vzhledem k vývoji hardware.
- **Problém**: programy nelze „nahackovat“, je potřeba **formálních metod** pro jeho specifikaci a ověření [ponecháno do předmětu navazujícího studia]
- **výuka principů** (ukázky v pseudokódu, konkrétní jazyk na cvičení)

Konkurentní (concurrent) programování

- „klasický“ program – sekvenční, (protože) instrukce jsou vykonávány procesorem sekvenčně
- **paralelní program** = „několik sekvenčních programů“ (procesů) **běžících současně** na samostatných procesorech
- paralelní programování = konstrukce paralelního programu
- **konkurentní program** = „několik sekvenčních programů“ (procesů), které **mohou být vykonávány současně**

= potenciálně paralelní programování – paralelizace může být zdánlivá, řešený sdílením zdrojů (procesoru)

- **konkurence** = abstrakce, předpokládání paralelního zpracování
- např. v OS obsluhy přerušení od hardware vykonávány konkurentně s programy, multitasking, multiprocesoring, distribuované systémy atd.
- **multithreading** v prog. jazycích – konkurentní vlákna v programu, např. UI a výpočet
- terminologie: proces vs. vlákno (thread)

Konkurentní (concurrent) programování

- procesy mohou (musí) **interagovat** a **komunikovat** $\Rightarrow$ potřeba je **synchronizovat**
- (mnohem) těžší než u sekvenčních programů docílit korektnosti a efektivnosti, chyby např. „zamrznutí“, „pády“ aplikací
- problémy **závislé v čase i situaci**, obtížné reprodukovat, diagnostikovat a opravit

Abstrakce konkurentního programu

- **konkuretní program** = (konečná) množina (sekvenčních) procesů
- procesy vykonávají (konečné) množství **atomických akcí**
  - dále nedělitelných = neproložitelných jiným procesem
  - izolovaných = dočasné stavy neviditelné
- **scénář (historie)** = posloupnost **libovolně proložených** atomických akcí procesů, zachováno pořadí akcí v rámci procesu
- následující atom. akce je **(nedeterministicky) vybrána** z následujících atom. akcí procesů, bez omezení (až na jednu výjimku)

Př. 2 procesy A a B s (neřídícími) akcemi $A1 \rightarrow A2$ a $B1 \rightarrow B2$. Možné scénáře:

Abstrakce konkurentního programu

- **konkuretní program** = (konečná) množina (sekvenčních) procesů
- procesy vykonávají (konečné) množství **atomických akcí**
  - dále nedělitelných = neproložitelných jiným procesem
  - izolovaných = dočasné stavy neviditelné
- **scénář (historie)** = posloupnost **libovolně proložených** atomických akcí procesů, zachováno pořadí akcí v rámci procesu
- následující atom. akce je **(nedeterministicky) vybrána** z následujících atom. akcí procesů, bez omezení (až na jednu výjimku)

Př. 2 procesy A a B s (neřídícími) akcemi $A1 \rightarrow A2$ a $B1 \rightarrow B2$. Možné scénáře:

$$A1 \rightarrow B1 \rightarrow A2 \rightarrow B2 \qquad B1 \rightarrow A1 \rightarrow B2 \rightarrow A2$$
$$A1 \rightarrow B1 \rightarrow B2 \rightarrow A2 \qquad B1 \rightarrow A1 \rightarrow A2 \rightarrow B2$$
$$A1 \rightarrow A2 \rightarrow B1 \rightarrow B2 \qquad B1 \rightarrow B2 \rightarrow A1 \rightarrow A2$$

$A2 \rightarrow A1 \rightarrow B1 \rightarrow B2$ není scénář. *Proč?*

- **synchronizace** = omezení počtu scénářů na nevedoucí k chybám, tzn. omezení výběru následující atom. akce

Zápis

Triviální konkurentní program				název programu
int n $\leftarrow$ 0				globální proměnné
				globální atom. akce
A		*B*		označní procesů
	int a $\leftarrow$ 1		int b $\leftarrow$ 2	lokální proměnné
1:	n $\leftarrow$ a	1:	n $\leftarrow$ b	atom. akce

Obrázek: Konkuretní program

Triviální sekvenční program
int n $\leftarrow$ 0
int a $\leftarrow$ 1
int b $\leftarrow$ 2
1: n $\leftarrow$ a
2: n $\leftarrow$ b

Obrázek: Sekvenční program

Zápis

Triviální konkurentní program				název programu
int n ← 0				globální proměnné
				globální atom. akce
A		*B*		označní procesů
	int a ← 1		int b ← 2	lokální proměnné
1:	n ← a	1:	n ← b	atom. akce

Obrázek: Konkuretní program

Triviální sekvenční program	
int n ← 0	
	int a ← 1
	int b ← 2
1:	n ← a
2:	n ← b

Obrázek: Sekvenční program

Stavy

- výpočet definován pomocí stavů a přechodů mezi nimi
- **stav** = n-tice aktuálních následujících atom. akcí procesů a hodnot globálních a lokálních proměnných
- přechody mezi stavy = vykonání následujících atom. akcí procesů
- **stavový diagram** = diagram (dosažitelných) stavů a přechodů mezi nimi

str. 11

Obrázek: Stavový diagram sekvenčního a konkurentního programu

Scénář (historie)

- **scénář** = posloupnost stavů dle přechodů mezi nimi, orientovaná cesta stavovým diagramem z počátečního stavu

A	B	n	$A.a$	$B.b$
1: n ← a	1: n ← b	0	1	2
konec	**1: n ← b**	1	1	2
konec	konec	2	1	2

Obrázek: Zápis scénáře

Paralelní architektury a abstrakce

- **multitasking** (time-sharing): 1 procesor, plánovač, přepnutí kontextu procesů (uložení lokálních proměnných) kdykoliv, tedy je možné lib. proložení instrukcí
- **multiprocesoring**: globální sdílená a lokální paměťi procesorů, (skutečně) paralelní vykonávání instrukcí, při práci s lokální pamětí nerozlišitelné od proložení instrukcí, u globální není přípustný paralelní přístup a je možné lib. pořadí procesorů, tedy i proložení instrukcí
- libovolné proložení atom. akcí = **ignorování času akcí a mezi akcemi**, umožněna formální analýza a nezávislé na aktuálním hardware, software a podmínkách běhu (!), důležité pouze pořadí (spočetného množství) akcí, které může být libovolné
- interakce procesů pomocí **globální sdílené paměti** (obecně sdíleného zdroje)

Distribuované programování

- **distribuovaný systém**: více procesorů (počítačů), žádná globální sdílená paměť, **posílání zpráv** mezi procesory (uzly) skrze kanály (orient. hrany)
- (skutečně) paralelní vykonávání atom. akcí, je možné lib. proložení (jen zpráva je před přijetím odeslána)
- **není globální sdílená paměť**, uzel nemůže přímo poslat zprávu všem ostatním, potřeba uvažovat **topologii uzlů**
- studium chování při chybách uzlů – „obejití“ uzlu
- [ponecháno do předmětu navazujícího studia]