Konzistentnost

Přednášky z distribuovaných systémů

Pro a proti replikaci

1. Zvýšení spolehlivosti.
2. Zvýšení výkonnosti.
3. Nutnost zachování škálovatelnosti systému co do počtu komponent i geografické rozlehlosti.
4. Výkonnost se snižuje, protože s opravou jedné kopie souvisí i oprava ostatních kopií pro zachování konzistentnosti.

Vztah ke škálovatelnosti (1)

- Pro zlepšení škálovatelnosti jsou používány replikace a cache.
- Škálovatelnost je obecně prezentována jako problém průchodnosti.
- Replikace dat v blízkosti potenciálních uživatelů může zlepšit výkonnost a zlepšit škálovatelnost.

Potenciální problémy:

- Šířka pásma požadovaná pro aktualizaci kopií může být velká.
- Udržování konzistentních replikovaných kopií může být problémové z pohledu škálovatelnosti.

Proto je třeba **dodržovat určité dohody při replikaci dat**.

Vztah ke škálovatelnosti (2)

- Vícenásobné kopie.
  - zvyšují výkonnost a redukují dobu přístupu
  - zvyšují také režii pro udržení konzistentnosti
  - příklad: N - krát replikované objekty
    - frekvence čtení R, frekvence zápisu W
    - je-li R <<W vysoká konzistentnost zvyšuje režii, zprávy jsou zbytečné
- Zdrojem řešení je udržování konzistentnosti
  - volba vhodné sémantiky
  - těsná konzistentnost vyžaduje globálně synchronizované hodiny
- Řešení: snížíme požadavky na konzistentnost
  - jsou k dispozici různé stupně konzistentnosti

Úvod do konzistenčních modelů

K popisu a řešení problému konzistentnosti bylo vytvořeno mnoho modelů, ve kterých se mluví o operacích **čtení** a **zápisu** nad **distribuovanou datovou pamětí**.

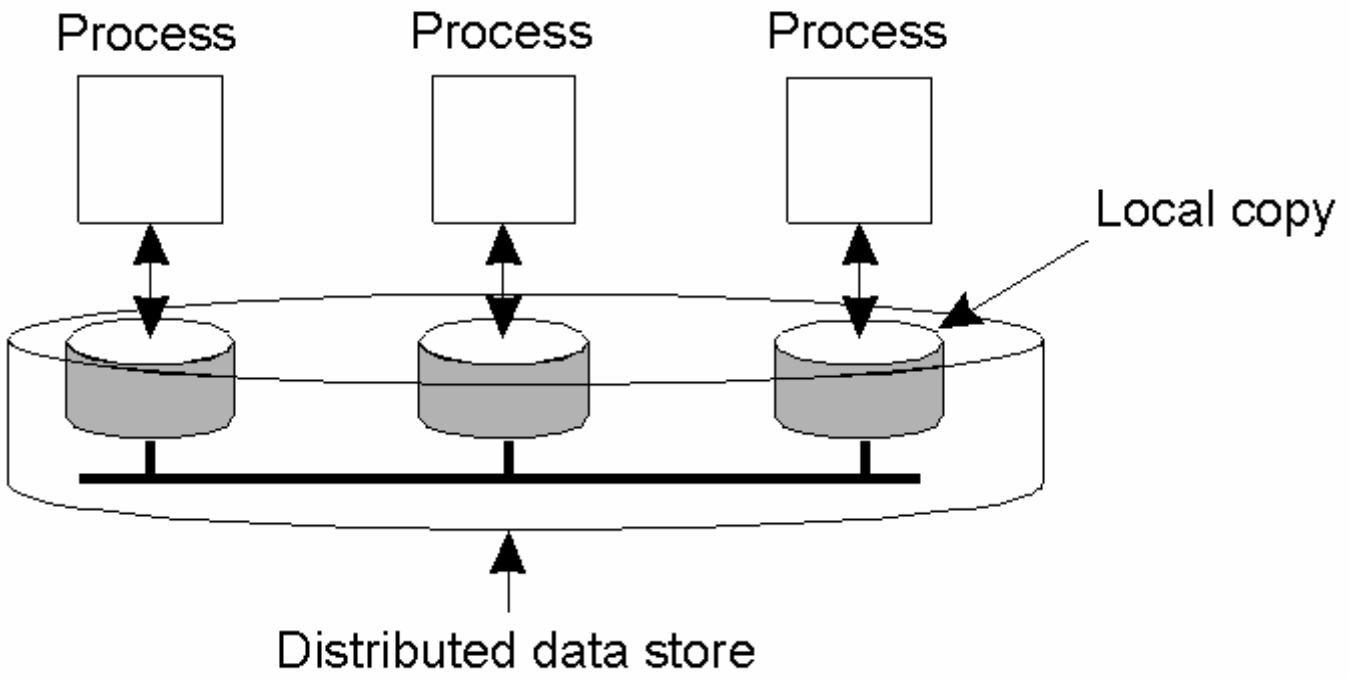
Každý proces je chápán tak, že má k dispozici kopii celé datové paměti.

Zápis je nějaká změna lokální kopie, kterou je třeba replikovat do ostatních vzdálených kopií.

Čtení je operace, která data nemodifikuje. Běžně je operace čtení chápána tak, že vrací výsledek poslední operace zápisu do datové položky.

Konzistenční model (konzistenční sémantika) je dohoda mezi procesy a datovou pamětí. Pokud budou procesy dodržovat dohody dané modelem, bude paměť pracovat korektně.

Data-Centric konzistenční modely

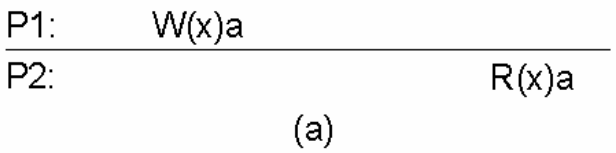


Obecná organizace paměti dat, fyzicky distribuované a replikované několika procesy.

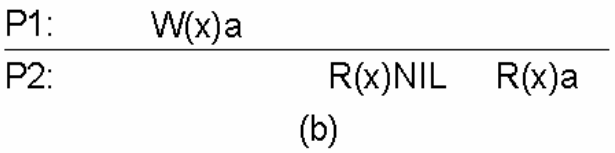
Striktní konzistentnost

Definice: jakékoliv čtení datové položky X vrací hodnotu odpovídající výsledku poslední operace zápisu X.

Definice implicitně předpokládá existenci absolutního globálního času. Je přirozeně dostupná v jednoprocesorových systémech, ale neimplementovatelná v distribuovaných systémech.



(a)



(b)

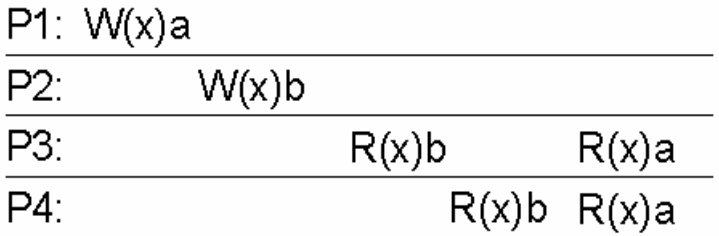
Dva procesy, pracující s týmiž daty.

a) Striktně konzistentní paměť.

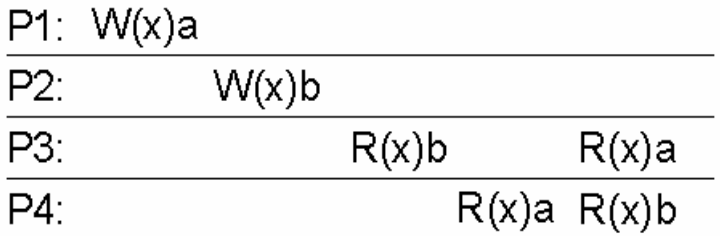
b) Paměť, které není striktně konzistentní.

Linearizovatelnost a sekvenční konzistentnost (1)

Sekvenční konzistentnost: Výsledek jakéhokoliv provádění je tentýž, jako kdyby operace čtení a zápisu všech procesů nad datovou pamětí byly prováděny v nějakém sekvenčním pořadí a operace všech individuálních procesů se jeví v téže sekvenci, ve které je specifikována jejich programem. Všechny procesy vidí totéž pořadí operací zápisu.



(a)



(b)

a) Sekvenčně konzistentní paměť.

b) Paměť, která není sekvenčně konzistentní.

Linearizovatelnost

- **Definice:** Výsledek jakéhokoliv provádění je tentýž jako kdyby operace čtení a zápisu všech procesů nad datovou pamětí byly prováděny ve stejném pořadí a operace všech individuálních procesů odpovídají v této sekvenci pořadí, specifikovaném jejich programem.Navíc pokud platí, že $TS(op_1(x)) < TS(op_2(y))$, pak operace op1(x) musí nastat dříve, než operace op2(y).
- V tomto modelu se předpokládá, že operace přijímají časové značky z globálně dostupného časového zdroje s konečnou přesností.
- Linearizovatelná datová paměť je také sekvenčně konzistentní, ale je implementačně náročnější než sekvenční konzistentnost.
- Linearizovatelnost se primárně používá při formální verifikaci konkurentních programů.

Linearizovatelnost a sekvenční konzistentnost (2)

Process P1	**Process P2**	**Process P3**
x = 1;	y = 1;	z = 1;
print (y, z);	print (x, z);	print (x, y);

Tři souběžně běžící procesy.

Linearizovatelnost a sekvenční konzistentnost (3)

x = 1;	x = 1;	y = 1;	y = 1;
print ((y, z);	y = 1;	z = 1;	x = 1;
y = 1;	print (x,z);	print (x, y);	z = 1;
print (x, z);	print(y, z);	print (x, z);	print (x, z);
z = 1;	z = 1;	x = 1;	print (y, z);
print (x, y);	print (x, y);	print (y, z);	print (x, y);
Prints: 001011	Prints: 101011	Prints: 010111	Prints: 111111
Signature: 001011 (a)	Signature: 101011 (b)	Signature: 010111 (c)	Signature: 111111 (d)

Čtyři platné posloupnosti provádění procesů z předchozího obrázku.

Sekvenční konzistentnost a serializovatelnost

- Sekvenční konzistentnost je porovnatelná se serializovatelností v případě transakcí.
- Odlišnost je v úrovni rozlišení: sekvenční konzistentnost je definována v termínech operací čtení a zápis, serializovatelnost v termínech transakcí, které zahrnují tyto operace.
- Sekvenční konzistentnost představuje uživatelsky přívětivý model, ale má vážné problémy s výkonností. Proto byly navrženy slabší modely konzistentnosti.

Příčinná (Casual) konzistentnost (1)

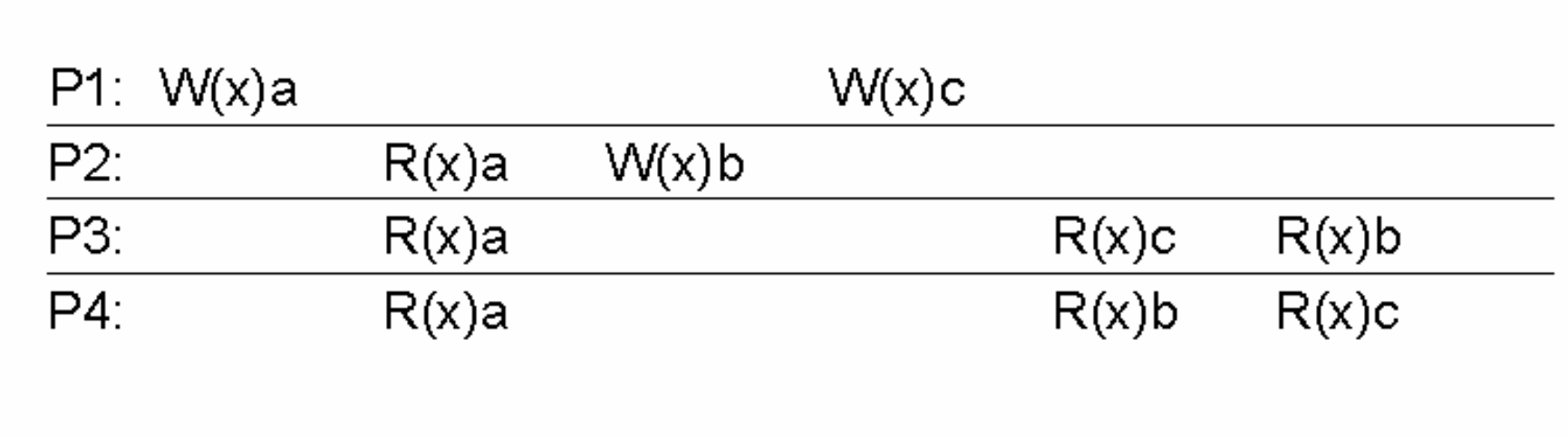
Příčinná konzistentnost vyžaduje úplné uspořádání pouze pro operace zápisu, které jsou vzájemně závislé.

1. Operace ***read*** je příčinně vztažena k operaci ***write***, pokud ***write*** zapisuje data, která ***read*** čte.
2. Operace ***write*** je příčinně vztažena k operaci ***read***, která nastane dříve než ***write*** v tomtéž procesu.
3. Jestliže ***read*** závisí na ***write_1*** a ***write_2*** na ***read***, pak také ***write_2*** závisí na ***write_1***.

Nutné podmínky:

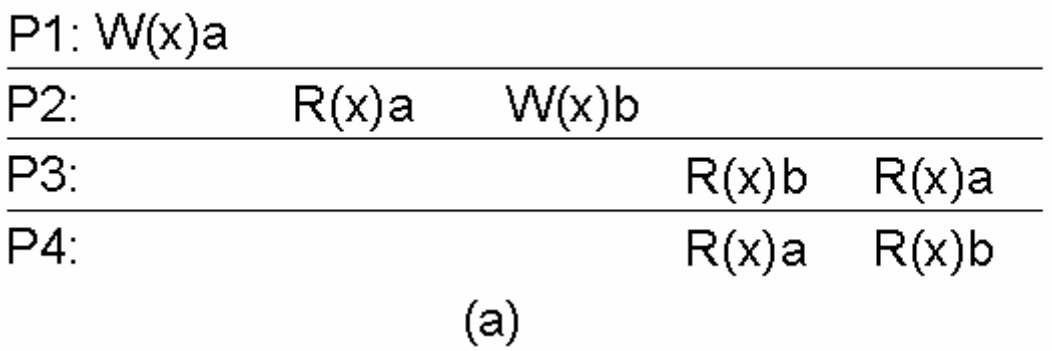
1. Zápisy, které mají potenciálně příčinný vztah musí být viděny všemi procesy ve stejném pořadí.
2. Konkurentní zápisy mohou být viděny v různých pořadích v různých pořadích v různých procesech.

Příčinná (Casual) konzistentnost (2)

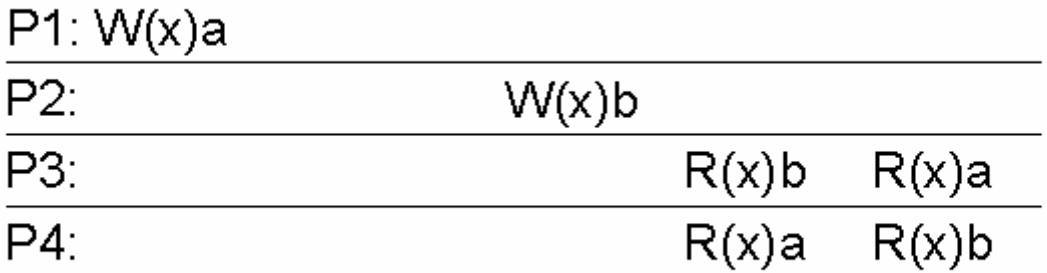


Tato sekvence je přípustná s příčinně konzistentní pamětí dat, ale ne jako sekvenčně nebo striktně konzistentní.

Příčinná (Casual) konzistentnost (3)



(a)



(b)

a) Porušení příčinně konzistentní paměti.

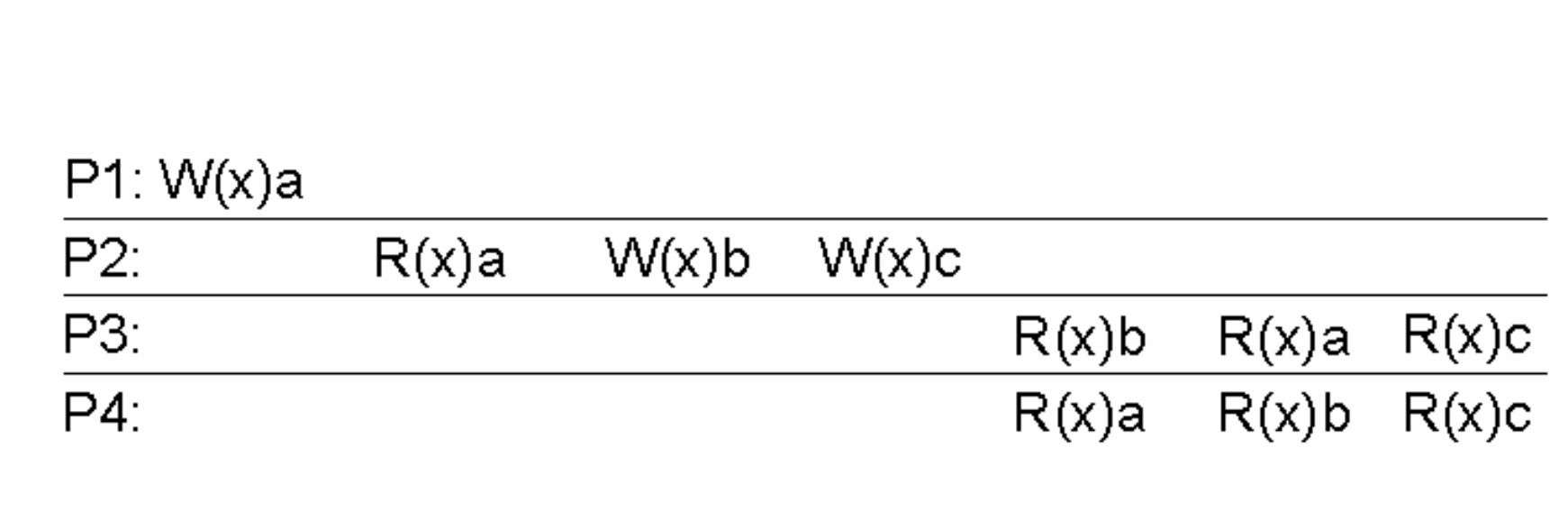
b) Správná sekvence událostí v příčinně konzistentní paměti.

FIFO konzistentnost (1)

Nezbytné podmínky:

1. Zápisy od jednoho procesu jsou ostatními procesy viděny v tomtéž pořadí, ve které byly vyslány
2. Zápisy od různých procesů mohou být viděny různými procesy v různém pořadí.

FIFO konzistentnost (2)



Platná sekvence událostí FIFO konzistentnosti

FIFO konzistentnost (3)

Process P1	**Process P2**	**Process P3**
x = 1;	y = 1;	z = 1;
print (y, z);	print (x, z);	print (x, y);

Tři souběžně běžící procesy.

FIFO konzistentnost (4)

x = 1;	x = 1;	y = 1;
print (y, z);	y = 1;	print (x, z);
y = 1;	**print(x, z);**	z = 1;
print(x, z);	print (y, z);	**print (x, y);**
z = 1;	z = 1;	x = 1;
print (x, y);	print (x, y);	print (y, z);
Prints: 00	Prints: 10	Prints: 01
(a)	(b)	(c)

FIFO konzistentní pohledy tří procesů na pořadí provádění příkazů z předchozího obrázku. Zvýrazněné příkazy **print** generují výstup.

FIFO konzistentnost (5)

Process P1	Process P2
x = 1;	y = 1;
if (y == 0) kill (P2);	if (x == 0) kill (P1);

Synchronizace dvou konkurentních procesů. Při FIFO konzistentnosti dávají různé výsledky (běží dva procesy, jeden proces nebo žádný proces).

Slabá (Weak) konzistentnost (1)

„Synchronizační proměnné” zajišťují konzistentnost skupiny operací (transakcí), ne individuálních zápisů a čtení.

Vlastnosti:

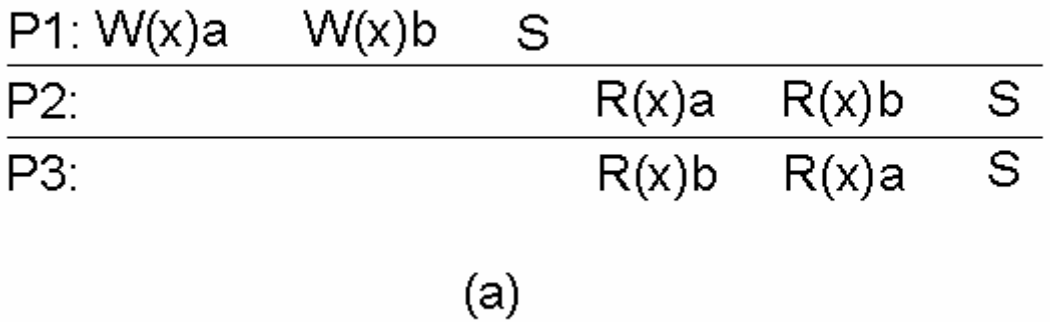
1. Přístup k synchronizačním proměnným spojeným s datovou pamětí je sekvenčně konzistentní
2. Není dovoleno, aby byla nad synchronizační proměnnou provedena operace zápisu, dokud není na všech místech dokončena předchozí operace zápisu
3. Není dovolena operace čtení nebo zápisu nad daty, dokud nejsou provedeny všechny předchozí operace nad synchronizačními proměnnými.

Slabá (Weak) konzistentnost (2)

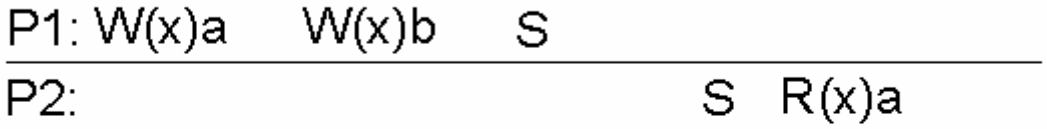
```
int a, b, c, d, e, x, y;          /* variables */
int *p, *q;                       /* pointers */
int f( int *p, int *q);           /* function prototype */

a = x * x;                        /* a stored in register */
b = y * y;                        /* b as well */
c = a*a*a + b*b + a * b;          /* used later */
d = a * a * c;                    /* used later */
p = &a;                           /* p gets address of a */
q = &b                            /* q gets address of b */
e = f(p, q)                       /* function call */
```

Část programu, kde mohou být některé proměnné uloženy v registrech.

Slabá (Weak) konzistentnost (3)



(a)

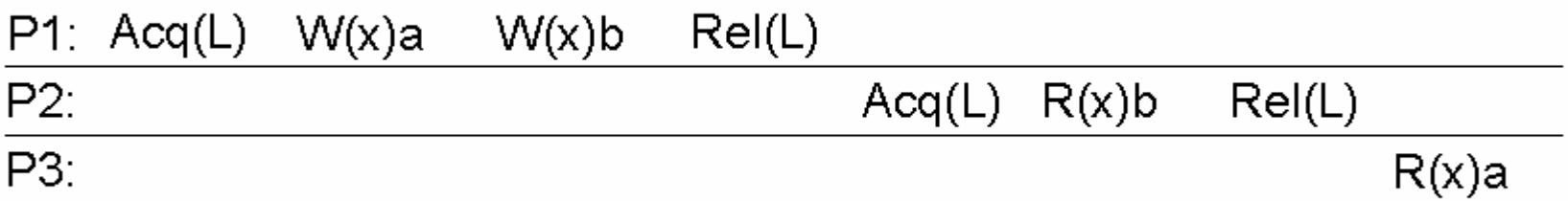


(b)

a) Platná posloupnost událostí pro slabou konzistentnost.

b) Neplatná posloupnost pro slabou konzistentnost.

Uvolňovaná (Release) konzistentnost (1)



Platná posloupnost událostí pro uvolňovanou konzistentnost.

Uvolňovaná (Release) konzistentnost (2)

Pravidla:

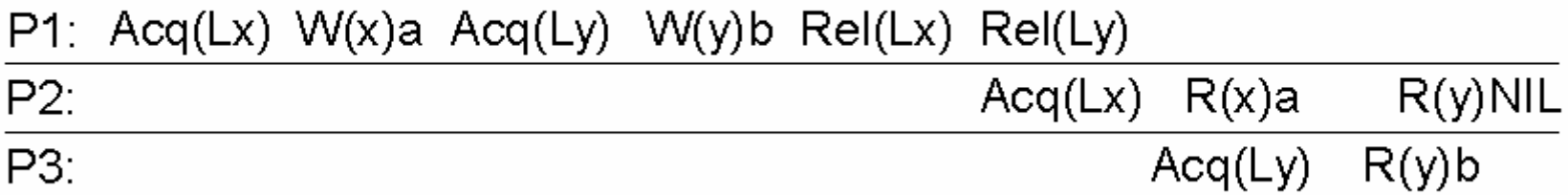
- Před provedením operace čtení nebo zápisu nad sdílenými daty musí být úspěšně ukončeny všechny předchozí požadavky
- Před provedením požadavku uvolnění (release), musí být ukončeny všechny předchozí čtení a zápisy daného procesu
- Přístupy k synchronizačním proměnným jsou FIFO konzistentní (sekvenční konzistentnost není požadována).

Vstupní (Entry) konzistentnost (1)

Podmínky:

- Získání přístupu k synchronizačním proměnným vztaženým k procesu je požadováno teprve tehdy, když jsou provedeny všechny opravy chráněných sdílených dat vztažených k tomuto procesu.
- Před povolením režimu výlučného přístupu k synchronizační proměnné procesem nesmí mít jiný proces přístup k této synchronizační proměnné ani v sdíleném (nevýlučném) režimu.
- Po ukončení režimu výlučného přístupu k synchronizační proměnné nemůže být proveden sdílený přístup k této synchronizační proměnné jinými procesy dokud není vzat na zřetel vlastníkem proměnné.

Vstupní (Entry) konzistentnost (2)



Platná sekvence událostí pro vstupní konzistentnost.

Přehled konzistenčních modelů

konzistentnost	popis
přísná	Absolutní uspořádání v čase pro všechny události sdílených přístupů.
linearizovatelnost	Všechny procesy musí vidět všechny sdílené přístupy ve stejném pořadí. Přístupy jsou navíc uspořádány podle globálních (nejednoznačných) časových značek.
sekvenční	Všechny procesy vidí všechny sdílené přístupy ve stejném pořadí. Přístup není uspořádán v čase.
příčinná	Všechny procesy vidí příčinně svázané sdílené přístupy v tomtéž pořadí.
FIFO	Všechny procesy vidí zápisy ostatních procesů v pořadí, ve kterém byly použity. Zápisy od různých procesů nemusí být viděny v tomtéž pořadí.

(a)

konzistentnost	popis
slabá	Sdílená data mohou být chápána jako konzistentní pouze po ukončení synchronizace.
uvolňující	Sdílená data jsou vytvořena konzistentně po ukončení kritické sekce.
vstupní	Sdílená data náležející kritické oblasti jsou vytvořena konzistentně po vstupu do kritické sekce.

(b)

a) Konzistentní modely, které nepoužívají synchronizační operace.

b) Modely se synchronizačními operacemi

Ostatní konzistenční modely (Klient-Centric konzistenční modely)

Předchozí studované modely se týkaly údržby konzistentní datové paměti v případě konkurentních operací zápisu a čtení.

Jiná třída distribuovaných datových pamětí je charakterizována tím, že postrádá souběžné opravy. Tj. mnoho operací požaduje čtení datové paměti, ale ne zápisy. Takové datové paměti používají velmi slabou formu konzistentnosti, známou jako ***možná, konečná (eventual) konzistentnost***.

Možná (Eventual) konzistentnost

Mezi systémy, kde může být velmi uvolněná konzistentnost patří:

- DNS systém
- Web.

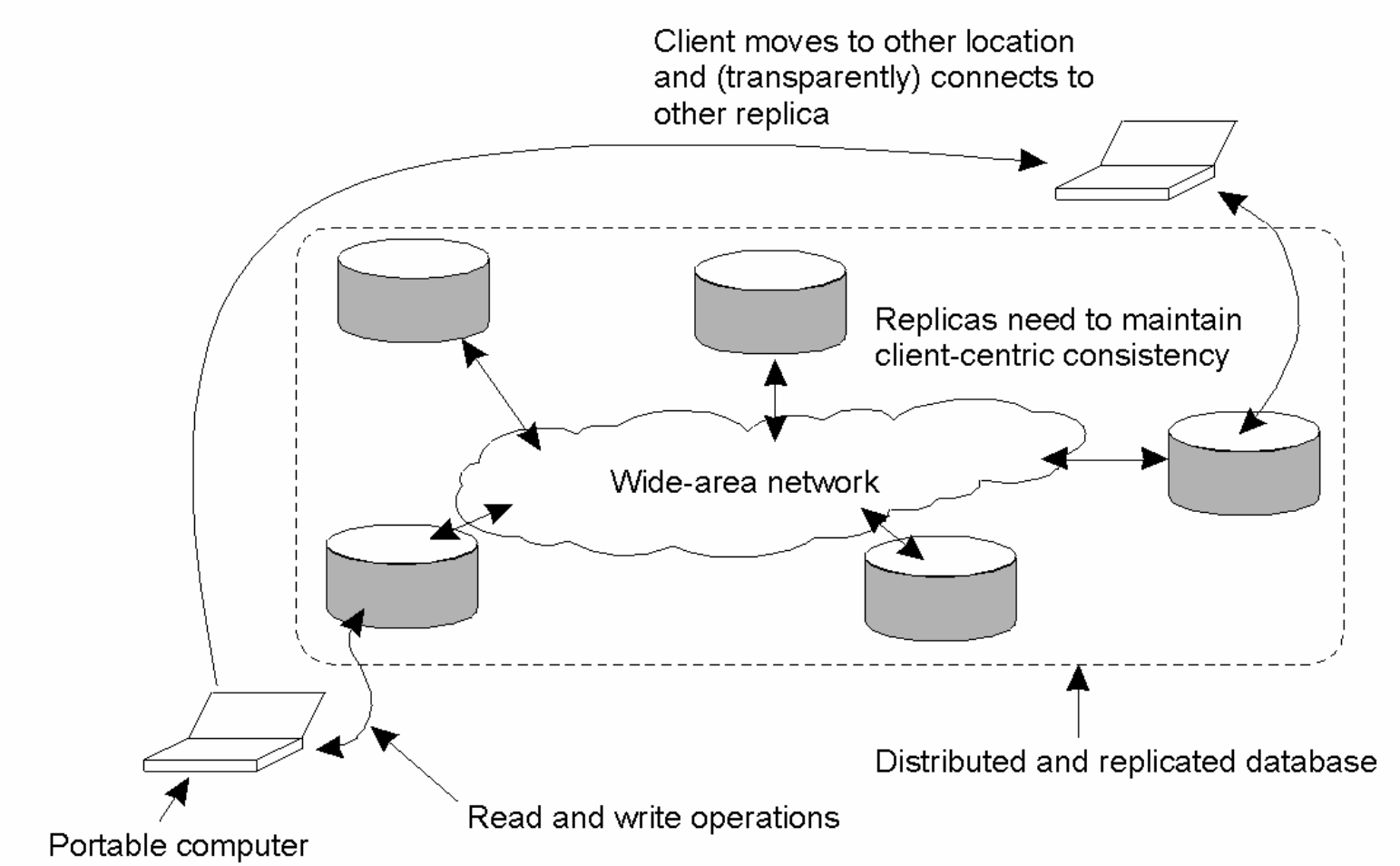
Otázka zní: jak rychle mohou být opravy přístupné procesům, které data pouze čtou?

Po nějaké době vidí všechny repliky všechny opravy a přechází do konzistentního stavu (jsou postupně konzistentní). Tento typ konzistentnosti je znám jako ***možná (konečná) konzistentnost***.

Možná (konečná) konzistentnost vyžaduje zadávat pouze takové opravy, u kterých je zaručeno, že budou v konečné době doručeny do všech replik.

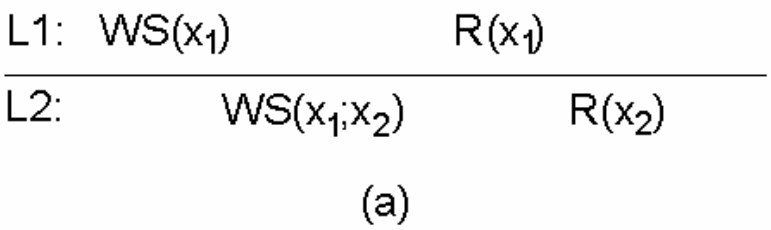
Možná (konečná) konzistentnost pracuje dobře pokud se klienti vždy připojují k téže replice. Pokud se do distribuovaných systémů přidá mobilita, může systém zkolabovat velmi rychle.

Možná (Eventual) konzistentnost

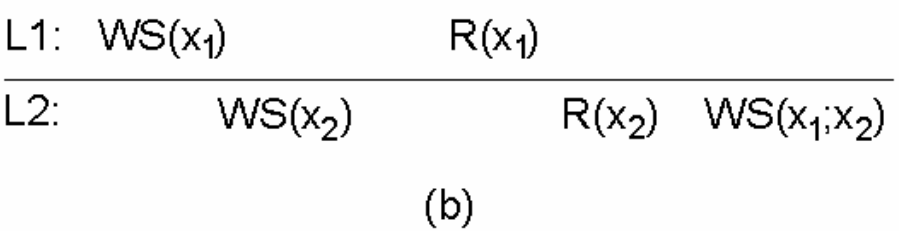


Princip přístupu mobilních uživatelů k různým replikám distribuované databáze.

Monotónní čtení



(a)



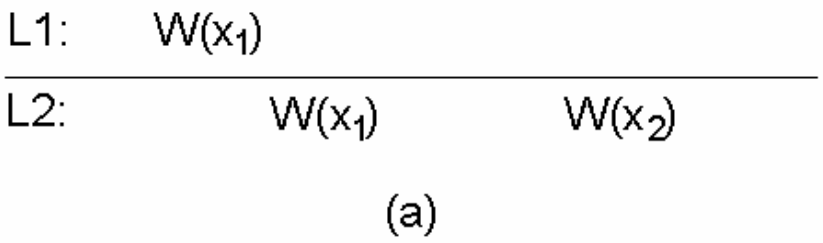
(b)

Operace čtení jsou prováděny jedním procesem *P* na dvou různých lokálních kopiích téže datové paměti.

a) Monotónní čtení konzistentní datové paměti

b) Datová paměť, kde se neprovádí monotónní čtení.

Monotónní zápis



(a)

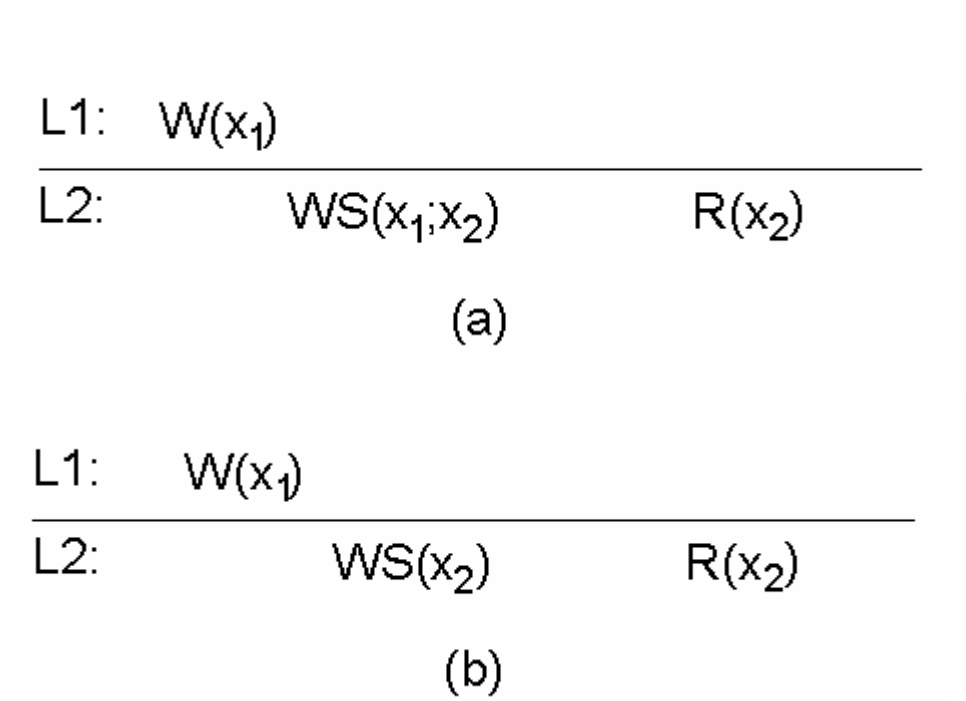
(b)

Operace zápisu prováděné jedním procesem *P* nad dvěma různými lokálními kopiemi téže datové paměti

a) Monotónní zápis konzistentní datové paměti

b) Datová paměť, která neprovádí konzistentní monotónní zápis.

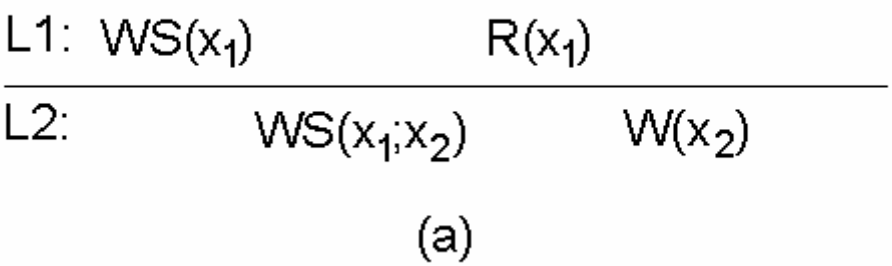
Čtení vlastních zápisů



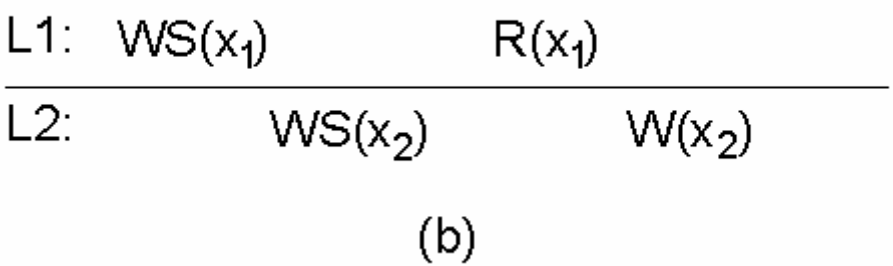
a) Datová paměť, která poskytuje konzistentnost typu čtení vlastních zápisů.

b) Datová paměť, která to neposkytuje.

Zápisy následující po čtení



(a)



(b)

a) Datová paměť s konzistentností typu „writes-follow-reads“.

b) Datová paměť bez konzistentnosti typu „writes-follow-reads“.