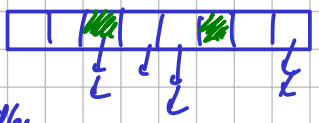
Paralelní DS

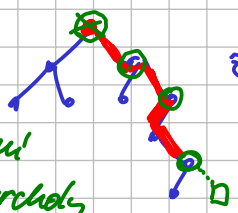
- Zámky (mutex): Lock, Unlock
  - 1 per instance
  - 1 per prvek ???
  - hešovací tabulka:



zámky pro přihrádky

přehešování ? R/W mutex

- vyhledávací strom

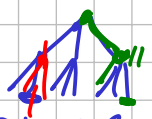


- Insert bez vyvažování
  - zamykám 2 vrcholy pod sebou

zamykání cest → pomalé

- (a,2a)-stromy

Insert



1 vrchol má mezi a-1 a 2a-1 klíči

(2,4)-strom → 1 až 3 klíče

Cvičení: Delete shora dolů

Příklad: spoj. seznamy (obousměrné)
atomický přesun prvku mezi SS.

A.Lock
B.Lock
přesun
B.Unlock
A.Unlock

B.Lock
A.Lock

AB-BA deadlock

Globální lin. uspoř. na zámcích, zamykám podle něj.

shora dolů: preventivně štěpit vrcholy s 2a-1 klíči

2a-1 → a-1 + a-1 + 1 (nové vrcholy / do otce)

Inv: aktuální vrchol má max. 2a-2 klíčů

stačí zamknout akt. vrchol a jeho otce

Problémy se zámky:
- riziko deadlocků
- férovost ?
- inverze priorit
- výkon
- tolerance k chybám

Rychlý
Pomalý

} DS Mutex

Atomické operace

- atomické registry
- exchange
- test and set bitu
- fetch & add
- paralelní přiřazení do n registrů
- LL/SC load locked, store conditional
  x←[adresa] [adresa]←y
- CAS compare & swap CAS(adresa, x, y)

1 z {

Korektnost: linearizovatelnost

Lin na operacích s objektem

int64 x
x ← 123456789 0x2

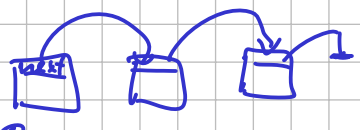
32	32

```
atomic {
    old ← [adresa]
    if old == x: [adresa] ← y
    return old
}
```

Příklad: Zásobník bez zámků

Node: atomic Node *next
... data
atomic int ref

Global: atomic Node *head

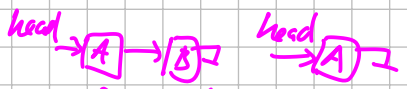


Push(Node *n):
loop{
h ← head
n.next ← h
if CAS(&head, h, n) == h:
return
}

Pop:
loop{
h ← head
HP ← h
if h ≠ head: continue
n ← h.next
if CAS(&head, h, n) == h:
return h
h.ref--
HP ← ∅
h.ref++
if h ≠ head:
h.ref--; continue
h.ref--
HP ← ∅
}

① je to lineární? Tak trochu...

② je to korektní? Ne:



Pop → A
Pop → B
Push(A)

ABA problém

Řešení: a) LL/SC místo CASu
b) double-CAS (CAS2) ← MC68020+
c) wide-CAS (WCAS) ← DCAS
d) omezíme recyklaci

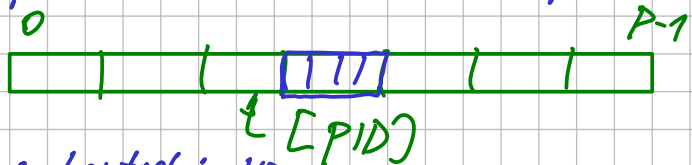
if (CAS2(<&head, &h.next>, <h, n>, <n, n>) == <h, n>): return

verzování pointerů → co když verze přeteče? reálně ignorujeme

if (WCAS(<&head, &version>, <h, ver>, <n, ver+1>))
...

③ Správa paměti

Řešení: a) free list + uvolnit časem → synchronizační body
b) reference counting → omezená recyklace paměti (na jiný účel)
trik: amortizace průchodů free-listem
pokud # procesů ≤ P
prvků referencovaných procesem ≤ R
pak # ref. prvků ≤ R·P
⇒ free-list procházím, když má ≥ 2·R·P prvků
c) hazard pointery
Safe Memory Reclamation



průchod free-listem: kontroluje HP