Datové struktury 1 z ZS 25/26

12. přednáška na Tři krále

Pavel Veselý

Plán: paralelní DS

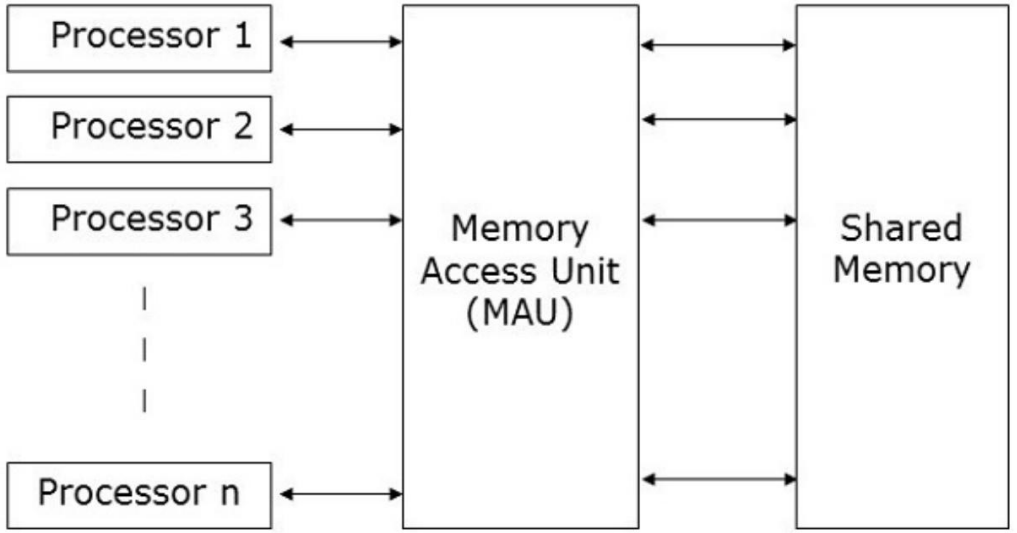
- Zámky a problémy s nimi
  - Použití na vyhledávací stromy

- Bezzámkové DS pomocí atomických operací
  - Zásobník
  - Problémy s dealokací

- Bonus: zlomkové kaskádování pro zrychlení Range trees z minula

Model Parallel RAM (Random Access Machine)

- m procesorů / vláken (instance RAMu)
- Paměť:
  - Lokální pro každé vlákno
  - Globální, sdílená
    - Složena z atomických buněk
    - Buňku je potřeba načíst do lokální paměti



- Konkrétní model: Concurrent Read Exclusive Write (CREW)
  - Umožníme souběžné čtení buňky více vlákny
  - Více souběžných zápisů – nedefinované chování
    - Podobně kombinace čtení a zápisů
- Rozcvička: sdílené počítadlo

➔ jsou potřeba synchronizační primitiva / atomické operace

Zámky (mutex = mutual exclusion)

- Synchronizační primitivum: zabraňuje souběžným operacím s jednou (částí) DS více vlákny
- Operace:
  - `Lock(L)`: Dokud L je uzamčen, čeká a pak zamkne L
  - `Unlock(L)`: Odemkne L (měl by být předtím zamknutý)
- Granularita použití:
  - Globální zámek
  - Zamknout celou instanci DS – velké „sekvenční zpomalení“
  - Zamykat části DS – vyžaduje více paměti
    - Např. přihrádky v hešovací tabulce
    - Nebo uzel ve stromu…

Zámky a problémy s nimi

1. **Deadlock** = vlákna navzájem čekají na odemčení
   - ∃ cyklus v orientovaném grafu čekání na zámky
     - Vrcholy = vlánka, hrana (p, q) = p čeká na zámek držený q
   - Řešení: uspořání na zámcích a zamykat v pořadí
     - Ale: ne vždy víme dopředu, co chceme zamknout …
2. Spravedlnost: kdo získá uvolněný zámek?
   - Vlákno může čekat neomezeně dlouho…
3. Nedodržení priorit procesů
   - Proces vyšší priority může čekat na proces nižší priority
4. Zpomalení programu + stojí paměť
5. Neuvolnění zámku při selhání vlákna

Dobrovolný DÚ:
Zahrajte si
Deadlock
Empire

Zamykání ve vyhledávacích stromech

- Zamykáme jednotlivé uzly zvlášť

Možnosti:

1. Zamknout postupně celou cestu do hledaného/nového uzlu
   - Vždy zamykáme kořen → stejné jako zamykat celou DS
2. Posouvat okénko o dvou uzlech po této cestě
   - Problém: vyvažování po `Insertu`
   - `Delete` nelze
3. Top-down verze (a, b)-stromů

Paralelizace top-down verze (a, b)-stromů

- Pouze pro $b \geq 2a$
- `Insert` jedním průchodem z kořene dolů
  - Pokud má aktuální uzel $b - 1$ klíčů, preventivně rozdělíme
    - Výsledné uzly mají $\frac{b-2}{2} \geq a - 1$ klíčů + jeden přidáme do rodiče
- `Delete` jedním průchodem z kořene dolů
  - Pokud má aktuální uzel $a - 1$ klíčů
    - Provedeme půjčení od bratra nebo spojení s bratrem

pro `Insert` stačí zamykat jen aktuální uzel + rodiče

- `Delete` složitější:
  - potřebujeme zamykat i bratra (levého nebo pravého)
  - Mazání mimo nejnižší vrstvu též potřebuje nahrazení prvku za následníka

Jak dokázat korektnost paralelních DS?

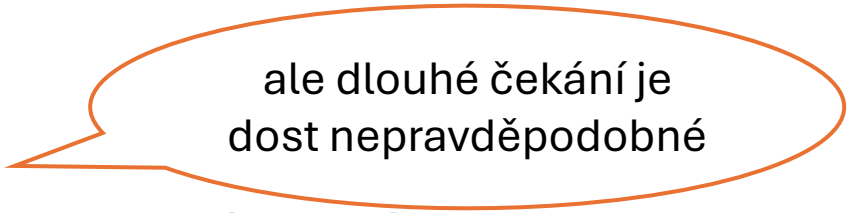
- Serializovatelnost DS:
  - ∃ lineární uspořádání O na operacích t.ž.:
    - Výsledek operace je konzistentní s předchozími operacemi v O
    - Každý proces vykonává své operace v pořadí konzistentním s O
  - Jaké pořadí operací zvolit pro top-down verzi (a, b)-stromů?

Bezzámkové DS pomocí atomických instrukcí

- `Read / write:` čtení / zápis sdílené atomické buňky
- `Exchange:` prohození obsahu lokální a sdílené buňky
  - Stačí na implementaci zámků 🔒
- `Fetch and add:` přičte k buňce danou hodnotu a vrátí původní
- `Compare and swap:` **CAS**(P, h, n): Pokud $P = h$, nastaví $P \leftarrow n$, a vždy vrátí původní hodnotu P
- `Load linked and store conditional` (**LL/SC**):
  - LL načte hodnotu sdílené buňky P do lokální proměnné L
  - následné SC zapíše hodnotu z L do P …
  - … ale jen pokud nedošlo k jinému přístupu k P, jinak selže

Bezzámkový zásobník

- Zásobník spojovým seznamem (uzel má hodnotu a pointer next)
  - Paralelizace přes CAS
- Problémy:
  - Livelock: vlákno může donekonečna cyklit
  - Problém ABA: prvek odebereme a pak vrátíme na zásobník
    - Řešení:
      - Použít LL/SC
      - Wide CAS: CAS na dvou sousedních buňkách
      - V průběhu Push alokovat nový uzel



Problémy s dealokací paměti

- Např. po operaci Pop bychom chtěli uzel dealokovat
  - Ale jiná operace na něj pořád může mít ukazatel
- Řešení: „free list“ F: seznam uzlů k uvolnění…
- … ale kdy provést samotné uvolnění paměti?

1. Globální synchronizace všech vláken
   - Všechny procesy se zbaví ukazatelů a pak můžeme vše v F uvolnit
2. Počítání referencí – každý uzel má počítadlo ukazatelů
   - Uvolňujeme uzly v F s nula referencemi
3. Hazardní ukazatele – každé vlákno drží seznam „hazardních ukazatelů“
   - Při uvolňování sebereme všechny hazardní ukazatele H
   - Uvolníme z F jen uzly, které nejsou v H

Hierarchie garancí paralelních DS

- Blocking: operace může čekat nekonečně dlouho (např. na zámek 🔒)
- Obstruction-free: operace se dokončí, pokud se ostatní vlákna zastaví
- Lock-free: alespoň jedno vlákno doběhne
  - pro některé ale může nastat livelock
- Wait-free: každá operace doběhne v konečném čase
- Bounded wait-free: každá operace doběhne v omezeném čase
  - např. omezeném funkcí počtu vláken

Jakou garanci má zásobník pomocí CAS?

Toť vše … teď ještě zkouška + navazující předměty

- Ústní s písemnou přípravou (max. 3 h, typicky méně) ~ ADS 1 a 2
- Teorie z přednášky
- Budeme zkoušet s Jiřím Finkem
- Jedna „větší“ a jedna „menší“ otázka
  - Při rozhodování o známce se můžeme zeptat i na něco jiného
- S sebou: psací potřeby, pití, dobrou náladu (volitelné)
  a nic elektronického
- Bc. studenti: pro uznání je potřeba známka ≤ 2
- Termíny a přihlašování v SISu
  - Pozor na rozestupy, tedy Váš přesný začátek
  - Předtermíny: též otázky z poslední přednášky (dám k tomu zdroje)