FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Semestrální práce z KIV/DS

TRANSAKCE

David Košek
A14N0132P
kosek@students.zcu.cz

1. Zadání

Databázový server obsluhuje paralelně několik klientů, kteří provádí v transakcích operace čtení/zápis nad daty ve sdíleném úložišti. Více klientů může zároveň v konkurentních transakcích přistupovat ke stejným položkám - server musí řešit přístupové konflikty.

- Implementace v C/C++.
- Překlad programem make.
- Žádné GUI, jen příkazový řádek, neinteraktivní - spustit, spočítat, zalogovat.
  OS Linux.
- Paralelizace - volitelně vlákna nebo procesy (dále v textu jen vlákna).
- Aplikace se sestává z 2 spustitelných programů - server a klient.
- Tabulka bude mít 100 řádků.
- Tabulka bude mít 3 sloupce:
  - identifikátor id - číslo řádku počítáno od 0, není tedy nutné, aby se implementovat,
  - hodnota typu int (celá čísla se znaménkem) - počáteční hodnota = id - 50 (tedy -50, 49, ..., 49),
  - časová značka posledního zápisu - počáteční hodnota 0, inkrementace po každém zápisu.
- Klient bude provádět operace podle zadaného dávkového souboru instrukcí.
- Druhy instrukcí:
  - BEGIN (začátek transakce)
  - COMMIT (spáchání transakce)
  - ADD *src1_id src2_id dst_id* (dst = src1 + src2, *_id jsou identifikátory řádků)
  - SLEEP *ms* (uspání klienta)
- Předpoklady:
  - Syntaxe a sémantika instrukční dávky bude vždy správná (žádné překlepy, spárované BEGIN a COMMIT, správné argumenty, ...).
  - Žádné vnořené subtransakce (transakce v transakci).
  - Instrukce ADD může být vykonána pouze v transakci (mezi instrukcemi BEGIN a COMMIT)
  - Instrukce SLEEP může být kdekoliv
  - V jedné transakci může být více instrukcí SLEEP i ADD.
- Komunikace se serverem probíhá jen při instrukcích BEGIN a COMMIT.
- Při instrukci BEGIN si klient vyžádá od serveru všechny hodnoty a jejich časové značky, které bude potřebovat v rámci této transakce.
- Při instrukci COMMIT klient odešle svá zpracovaná (změněná) data serveru společně s původními časovými značkami a získá od serveru odpověď, zda bylo spáchání úspěšně provedeno.

2. Implementace

2.1 Server

Při startu server se nastaví hodnota portu a IP adresy, na které server bude naslouchat. Server čeká na příchozí spojení s klientem. Pro každé připojení se startuje nové vlákno, které obsluhuje daného klienta po dobu jeho transakce. Nejprve přečte požadované řádky, které klient požaduje, které mu následně odešle s aktuální časovou značkou. Klient si je zpracuje, upraví a pošle zpět. Pokud jsou časové značky stejné, víme, že nám data nikdo nepřepsal a commit můžeme prohlásit za úspěšný. V opačném případě se jedná o abort.

2.2 Klient

Při startu nové klienta dojde k načtení dávkového souboru, který je následně zpracováván. Příkazem begin se připojím k serveru, který požádám o potřebné řádky, které chci změnit. Server mi vrátí požadovaná data, která upravím a pošlu zpět. Server mě poté informuje o tom, zda se transakce povedla, či ne.

3. Spuštění

3.1 Server

Server se spouští příkazem *./Server <port>*, kde port je číslo portu na kterém bude server naslouchat. IP adresa je 127.0.0.1 (lokální). Server jde jednoduše ukončit pomocí kláves CTRL+C. Po přeložení je spustitelný soubor ve složce Finish.

3.2 Klient

Klient se spouští příkazem *./Client <port> <id> <soubor_s_prikazy>*, kde port je číslo portu na kterém naslouchá server. Klient zpracuje dané příkazy a sám se ukončí. Po přeložení je spustitelný soubor ve složce Finish

3.3 Net flow vizu dia

Ve složce log jsou po ukončení serveru a klientu dostupné textové soubory se za logovanými hodnotami. Pro vygenerování grafu nám slouží skript create_dia.sh.

4. Závěr

Server i klienti pracují dle očekávání. Největší problém bylo správné za logování času přijatých a odeslaných zpráv. Komunikace mezi procesy ukazuje výsledný graf. Vidíme, že klient začne transakci a žádá o data, server mu data vrací, klient je upravuje a pokud se časová značka nezměnila, je transakce úspěšná.