ALGORITMI PARALELI ȘI DISTRIBUIȚI
Tema #1b - Acces concurent in baze de date

Responsabili: Silviu-George Pantelimon

Termen de predare: 06-12-2024 23:59 (soft), 13-12-2024 23:59 (hard)
Ultima modificare: 15-11-2024 12:45

Cuprins

Introducere

Acestă temă are ca scop familiarizarea studenților cu design pattern-ul de thread-pool și probleme asociate folosirii acestuia. În multe aplicații unde se dorește responsivitatea mare, cum sunt serverele, de obicei avem un fir de execuție principal care așteaptă cereri din rețea care sunt prelucrate, și se returnează un răspuns.

În contextul aplicațiilor concurente, serverul se poate folosi de mai multe fire de execuție, dintre care fiecare preia o cerere să o trateze. Astfel, se maximizează numărul de cereri ce pot fi prelucrate în paralel. Lucrurile însă se complică atunci când aplicația trebuie să fie *stateful*, adică se persistă o stare, de obicei într-o bază de date, iar aici avem condiții de cursă legate de accesul la baza de date.

Cu această temă, vom încerca să implementăm un thread-pool care să execute două operații pe o bază de date *in-memory*, de scriere și de citire, astfel încât starea bazei de date să fie consistentă. De asemenea, tema va trebui să implementeze rezolvarea problemei consistenței acestei baze de date în trei moduri, unul care prioritizează citirile, și două care prioritizează scrierile diferite (hint: aveți acces la monitorul din Java pentru a doua soluție).

Implementare

În scheletul din repository-ul temei, aveți deja implementată baza de date din memorie care reprezintă o listă de blocuri în care se scriu șiruri de caractere la un anumit index, care are asociat și un număr de secvență care se incrementează la fiecare scriere.

Scheletul de cod conține și testele care generează aleator o secvență de operații de scriere și citire. Indiferent de ordinea operațiilor și de datele prelucrate, baza de date trebuie să conțină aceleași valori date spre scriere. Voi trebuie să implementați cele trei soluții și să rulați testele conform input-ului dat în metoda *ExecuteWork* din clasa *TaskExecutor* folosind un thread-pool propriu implementat.

Veți trimite rând pe rând operațiile către thread-pool, iar firele de execuție trebuie să le proceseze corect în funcție de tipul operațiilor.

În cod există deja ca exemplu implementarea serială, pe care puteți să o transformați în varianta paralelizată, însă testele eșuează pentru acesta. Testele vor verifica atât corectitudinea implementării, cât și performanța acesteia.

Obiective

Obiectivele acestei teme sunt:

- Înțelegerea unui thread-pool prin implementarea de la zero a acestuia.
- Rezolvarea prin mai multe modalități a problemei cititori - scriitori pentru o înțelegere mai bună.
- Combinarea mai multor concepte din programarea paralelă și OOP pentru a avea o soluție elegantă. Există mai multe moduri prin care se poate implementa tema dar, ideal ar fi bine să aveți în minte că lucrați într-un limbaj OOP și că programarea paralelă este facilitată de acest lucru.
- Să aveți contact cu TDD (Test-Driven Development). Este o metodologie prin care la începutul dezvoltării se scriu testele care vor pica și apoi se implementează soluția care să treacă aceste teste.

Notare

Tema se poate testa local, direct în IntelliJ, sau folosind Docker, după cum se explică mai jos. Tema se va încărca pe Moodle, sub formă de arhivă Zip care, pe lângă fișierele sursă, va trebui să conțină următoarele fișiere **în rădăcina arhivei**:

- folderul "main" cu toată structura ca în schelet, adică cu ierarhia "main/java/org/apd" și fișierele din acesta.
- un fișier README

Punctajul total de 100 de puncte este împărțit în felul următor:

- **30p** - soluție cu prioritizare citiri (5p / test)
- **30p** - prima soluție cu prioritizare scrieri (5p / test)
- **30p** - a doua soluție cu prioritizare scrieri (5p / test)
- **10p** - README cu explicații (cu mențiunea că se ia în considerare doar dacă tema trece toate testele la cel puțin una din soluții).

În README, va trebui să explicați nu ce ați facut în cod, ci ce concluzii legate de performanță puteți trage din urma implementărilor, inclusiv observații legate de modul în care au fost făcute testele.

Restricții temă

Pentru rezolvarea temei aveți următoarele restricții:

- NU aveți voie să închideți și apoi să redeschideți fire de execuție după terminarea unor operații decât după ce s-au terminat de procesat toate operațiile. Implementarea trebuie să fie neapărat cu un thread-pool.
- NU aveți voie să folosiți ExecutorService din Java sau orice altă clasă care deja vă implementează un thread-pool. Implementarea trebuie să fie a voastră.
- NU puteți folosi aceiași implementarea pentru o rezolvare problemei cititori-scriitori ca să treacă testele pentru celelalte implementări.

Depunctare

Indiferent de trecerea testelor, punctajul obținut va fi complet și irevocabil anulat în următoarele situații:

- folosirea oricărui tip de inteligență artificială, care poate fi (dar nu se rezuma doar la) ChatGPT sau GithubCopilot, indiferent dacă este folosit pentru cod sau README
- copierea unei soluții parțial sau complet de la colegi
- prezentarea oricărei soluții seriale pentru trecerea testelor
- prezentarea oricărei soluții în locul alteia pentru trecerea testelor
- nefolosirea implementării proprii de thread-pool, inclusiv prin paralelizarea unui "for" pentru lista de cereri de scrieri și citire
- folosirea oricărei biblioteci care implementează un thread-pool.

De asemenea, se va anula doar punctajul pe README daca nu este suficient explicat ce ați putut deduce din implementare.

Atenție! Checker-ul vă dă cele 90 de puncte pentru implementare, dar acela nu este punctajul final (dacă, de exemplu, tema dă rezultate corecte, dar ați folosit o bibliotecă de thread-pool, veți avea nota finală 0).

Nu în ultimul rând, penalitatea cea mai devastatoare pentru cei ce încalcă cele menționate este prezența pe o listă neagră, unde se află studenți care în mod irevocabil nu vor mai putea face licența, dizertația și chiar și doctoratul cu Radu-Ioan Ciobanu. Abia cand veți vedea că cei care nu au încălcat normele noastre de etică se laudă că își fac aceste lucrări cu Radu, o să înțelegeți gravitatea lucrurilor, și o să vă simțiți prost pentru că ați ratat această ocazie. Așa că fiți ca Radu, cinstiți și muncitori, ca să puteți colabora cu el.

Testare

Pentru a vă putea testa tema aveți doua posibilități.

IntelliJ

Puteți folosi IntelliJ ca IDE și să rulați testele din fișierul *ExecutorTests* prin sageata verde din dreptul clasei sau să rulați teste individuale cu sageata verde din dreptul metodelor acestei clase. **Atenție!** Codul este de Java 21 (ca să vedeți că nu mai e chiar atât de învechit limbajul). Puteți descărca un JDK mai nou cu dublu-shift, căutați "Download JDK" și deschideți meniul de descărcare.

Docker

Pentru a avea un mediu uniform de testare, sau pentru a putea rula scriptul de test mai ușor de pe Windows sau MacOS (și cu mai puține resurse consumate decât dintr-o mașină virtuală), vă sugerăm să folosiți Docker. În repository-ul temei, găsiți un script numit *run_with_docker.sh* și *run_with_docker.bat* pentru Linux/OSX respectiv Windows. Dacă aveți Docker instalat și rulați acest script, se va rula testul automat în interiorul unui container. **Atenție!** Nu modificați fișierele din folderul *checker*.

Link-uri utile

1. Thread pool
2. Thread pool 2
3. Scriitori - cititori
4. Laboratorul 7