ISPITNI RAD

Kandidat: Relja Brdar
Broj indeksa: SV30/2023

Predmet: Paralelno programiranje
Tema projekta: MultiThreadScraper

Novi Sad, septembar, 2025.

---

1. Uvod

1.1 MultiThreadScraper

Cilj projekta je implementacija **paralelnog web skrejpera** koji preuzima informacije o knjigama sa sajta *Books to Scrape*. Program koristi **TBB (Threading Building Blocks)** i **std::thread** biblioteku za rad sa višestrukim nitima, dok se za HTTP zahteve koristi biblioteka **CPR**.

Skrejper obilazi sve dostupne stranice, prikuplja informacije o knjigama (naslov, cena, dostupnost, ocena, opis), kao i rezultate po kategorijama. Sistem meri statistike kao što su prosečna cena, minimalna i maksimalna cena, ukupan broj dostupnih knjiga i broj knjiga sa maksimalnom ocenom.

1.2 Zadatak

Cilj projekta je implementirati **paralelni skrejper** u C++ jeziku, sa sledećim funkcionalnostima:

- Paralelno preuzimanje HTML stranica korišćenjem višestrukih niti/ task grupa
- Sinhronizovan pristup zajedničkim strukturama podataka korišćenjem TBB kontejnera
- Ekstrakcija linkova i podataka pomoću regularnih izraza
- Računanje agregatnih statistika (min, max, prosek, broj dostupnih knjiga)
- Kreiranje izlaznog fajla sa rezultatima skrejpinga

---

2. Analiza problema

2.1 Izazovi problema

Glavni izazov u razvoju ovog projekta je **upravljanje konkurentnim pristupom podacima**. Više niti istovremeno obrađuje URL-ove, pa je neophodno obezbediti:

- Izbegavanje **race condition-a** prilikom upisa u globalne strukture (visited set, queue, statistike).
- Efikasnu sinhronizaciju bez velikih blokada niti.
- Upravljanje velikim brojem konekcija ka serveru bez gubitka performansi.

2.2 Razmatranje pristupa rešavanju problema

Paralelizam i radnici (workers)

Korišćen je model **work-stealing** – svaka nit uzima URL iz reda i obrađuje ga. Korišćen je `tbb::concurrent_queue` kao thread-safe red.

Sinhronizacija i bezbedne strukture podataka

Umesto standardnih kolekcija korišćeni su **TBB konkurentni kontejneri** (`concurrent_queue`, `concurrent_unordered_set`, `concurrent_vector`) koji sprečavaju race condition bez manuelnih mutex-a.

Arhitektura objektno-orijentisanog dizajna

Program je podeljen na module:

- **Worker** – obrada jednog URL-a (fetch + parsiranje + dodavanje novih linkova).
- **PageParser** – parsiranje HTML-a (knjige, kategorije, linkovi).
- **Structs** – globalne strukture i statistike.

Upravljanje memorijom

Korišćen je RAII pristup i TBB kontejneri, što eliminiše curenje memorije i olakšava paralelni rad.

Performanse i skalabilnost

Broj niti je postavljen na `hardware_concurrency() * 16` kako bi se maksimalno iskoristila paralelizacija. Testirano je na većem broju stranica i različitim brojevima niti - ovaj broj se pokazao kao “sweet spot”.

Upravljanje datotekama

Na kraju skrejpinga podaci se beleže u fajl `book_data.txt`, sažeto u vidu statistike i rezultata po kategorijama.

---

3. Rešavanje problema

3.1 Sekvencijalni program – moduli i osnovne metode

3.1.1 Glavni program (main)

- Inicijalizuje red linkova početnom URL adresom.
- Pokreće više niti (workers).
- Čeka završetak svih niti.
- Na kraju snima statistiku i rezultate u fajl.

3.1.2 Klasa Structs

- Sadrži **globalne strukture**: red linkova, skup posećenih URL-ova, rezultate kategorija, statistike.
- Implementira metod `update_price_stats` za ažuriranje min i max cena.

3.1.3 Klasa Worker

- Preuzima URL iz reda.
- Šalje HTTP zahtev (`fetch`).
- Prosleđuje HTML sadržaj parseru.
- Ubacuje nove linkove u red.

3.1.4 Klasa PageParser

- Ekstrakcija linkova iz HTML-a.
- Parsiranje stranice knjige (naslov, cena, dostupnost, ocena, opis).
- Parsiranje stranica kategorija i rezultata.

---

4. Testiranje

4.1 Testni skupovi

4.1.1 Test 1 – Testiranje skrejpinga početne stranice

- **Cilj:** Provera ispravnosti obrade početnog URL-a.
- **Rezultat:** Uspešno prepoznati linkovi, dodati u red i obrađeni.

4.1.2 Test 2 – Testiranje parsiranja knjiga

- **Cilj:** Provera ispravnog preuzimanja naslova, cene, ocene i dostupnosti.
- **Rezultat:** Podaci tačno upisani u `book_data`. Statistika ažurirana (prosečna cena, min, max).

4.1.3 Test 3 – Testiranje rada sa više niti

- **Cilj:** Provera korektne sinhronizacije kod 16 niti - `hardware_concurrency()` na mom računaru.
- **Rezultat:** Nema dupliciranja URL-ova, nema race condition-a. Performanse značajno bolje nego sa jednom niti.

4.1.4 Test 4 - Testiranje različitog broja niti/task grupa

- **Cilj:** Ispitivanje relacije između broja niti i vremena izvršavanja programa
- **Vrednosti parametara:** pokušano sa vredostima 16, 64, 256, 512 i 1024, koristeći vektor `std::thread` -ova ili `tbb::task_group`

- **Rezultati:** optimalan broj niti je između 256 i 512 dok beležimo značajan skok u vremenu rada pri korišćenju `tbb::task_group` -a nezavisno od broja taskova

---

5. Zaključak

Projekat **MultiThreadScraper** uspešno demonstrira upotrebu **paralelnog programiranja u C++-u** kroz praktičan problem web skrejpinga.

Korišćenje **TBB konkurentnih struktura** omogućilo je siguran i efikasan rad sa više niti, dok je organizacija u klase **Worker, PageParser i Structs** omogućila jasnu podelu odgovornosti.

Testiranjem je potvrđena ispravnost sistema i merenja pokazuju značajno ubrzanje sa povećanjem broja niti. Projekat predstavlja primer kako se **paralelizam i objektno-orijentisani dizajn** mogu uspešno kombinovati za rešavanje problema obrade velike količine podataka u realnom vremenu.