# Heap Manager

(Sistem za upravljanje memorijom)

Grupa 16

Studenti: Srđan Delić, Milan Tripković

## Uvod

Ovaj projekat implementira sistem za upravljanje memorijom sa ciljem simulacije rada heap menadžera u okruženju sa više niti. Problem koji se rešava je bezbedno i efikasno upravljanje dinamičkom memorijom kada više klijenata (ili niti) zahteva alokaciju i dealokaciju istovremeno. Poseban izazov predstavlja istovremena upotreba deljenih resursa i izbegavanje konkurentnog pristupa.

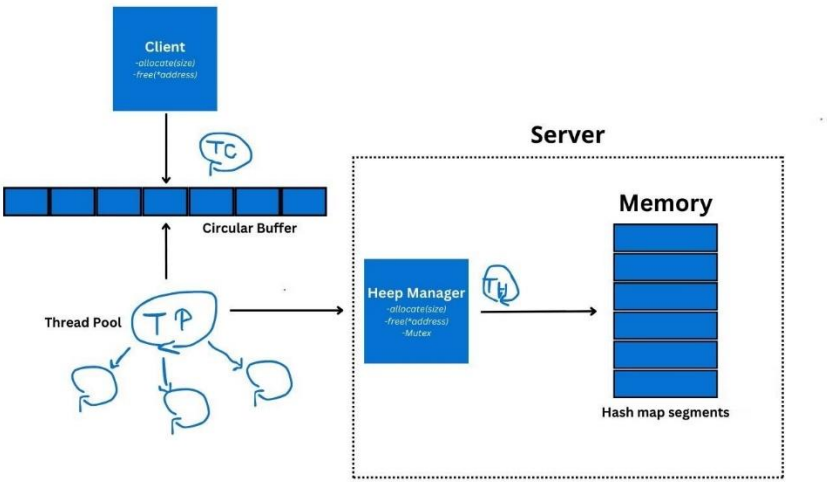
Ciljevi zadatka:

- Implementacija heap menadžera koji podržava konkurentan pristup.
- Podrška za algoritme FIRST FIT i NEXT FIT.
- Implementacija thread pool-a za obradu zahteva.
- Upotreba kružnog bafera za sinhronizaciju između niti.
- Vizualizacija stanja heap-a i sistema.

## Dizajn

Implementacija je modularna i sastoji se iz sledećih komponenti:

- **HeapManager**: logika za alokaciju i dealokaciju memorije (sa jednom ili više zona).
- **CircularBuffer**: kružni bafer za prenos zahteva između niti.
- **ThreadPool**: fiksni broj niti koji obrćuju zadatke.
- **Server**: mrežna komponenta za prihvat zahteva klijenata.



## Strukture podataka

- **Segment**: predstavlja jedan blok memorije, sadrži adresu, status i pokazivač na sledeći segment.
- **CircularBuffer**: bafer fiksne veličine, koristi semafore za konkurentan pristup.
- **ThreadPool**: struktura sa radnim nitima koje konstantno čekaju zadatke.
- **Stripe/Heap zone**: niz nezavisnih listi sa mutex-ima za paralelnu alokaciju.

**Razlozi izbora:**

- **Segmenti** su jednostavni za manipulaciju.
- **Kružni bafer** je efikasan za razmenu podataka bez aktivnog čekanja.
- **Thread pool** smanjuje troškove kreiranja i uništavanja niti.
- **Više zona (striping)** smanjuje zagušenje oko mutex-a.

## Rezultati testiranja

**Testovi obuhvataju:**

- Višestruki zahtevi za alokaciju i dealokaciju sa različitih niti.
- Simulacija rada sa 1000+ zahteva (stres test).
- Vizuelna provera preko getHeapOverview() i MONITORING komande.
- Konekcija 2 klijenta koji paralelno šalju 50 zahteva u sekundi.
- Praćenje ponašanja sistema putem monitoring interfejsa na klijentskoj strani.

**Monitoring obuhvata:**

- Trenutno stanje bafera.
- Broj aktivnih niti.
- Broj obrađenih zahteva.
- Maksimalno zabeleženi broj niti aktivnih u isto vreme.
- Maksimalno zauzeće bafera tokom testa.
- Dinamičko proširenje/smanjenje veličine bafera na osnovu broja zahteva u njemu.

**Rezultati:**

- Nije došlo do deadlock-a ni pri velikom opterećenju.
- Ispravno menjanje algoritma (FIRST FIT <-> NEXT FIT) na osnovu broja segmenata.
- Ispravno proširenje/smanjenje veličine bafera na osnovu broja zahteva u njemu.
- Tokom testa svi ključni događaji su evidentirani i snimani u `.txt` fajl radi kasnije analize.

## Zaključak

- Implementirano rešenje uspešno zadovoljava ciljeve zadatka – omogućava konkurentnu alokaciju i dealokaciju memorije sa minimalnim kašnjenjem i bez grešaka pri visokom opterećenju.
- Mehanizam prelaska sa **First Fit** na **Next Fit** algoritam se dinamički aktivira i smanjuje vreme pronalaženja slobodnih segmenata pri velikom broju alokacija.
- Dinamičko proširenje/smanjenje veličine bafera omogućava efikasno upravljanje memorijom u skladu sa brojem dolaznih zahteva, što je ključno za stabilnost sistema.
- Rezultati su **bolji od očekivanih** jer se sistem pokazao skalabilnim, stabilnim i brzim čak i pri velikim opterećenjima bez gubitka zahteva ili grešaka u radu.

## Potencijalna unapređenja

- **Implementacija vizuelnog interfejsa** za nadzor nad monitoringom sistema radi bolje preglednosti i upotrebljivosti.
- **Dodavanje prioriteta za zahteve** (npr. hitni vs. obični), čime bi se omogućila efikasnija raspodela resursa u realnim sistemima.
- **Naprednija strategija oslobađanja memorije**, npr. spajanje susednih slobodnih segmenata (coalescing), kako bi se sprečila fragmentacija.
- **Dinamičko proširivanje broja niti u thread pool-u** Trenutna implementacija koristi fiksni broj niti. U scenarijima kada sve niti postanu zauzete, novi zahtevi čekaju na obradu. Uvođenjem dinamičkog proširivanja (npr. ako je broj aktivnih niti konstantno na maksimumu određeni vremenski period), sistem bi mogao bolje da odgovori na povećano opterećenje.