

Факултет техничких наука

Индустријски комуникациони протоколи

Напредни алокатор меморије

Пројекат

Аутори:

Јована Мишић ПР 36/2022

Андијана Стојковић ПР 42/2022

Фебруар 2026.

Садржај

1. Увод
2. Дизајн
3. Циљ тестирања

1. Увод

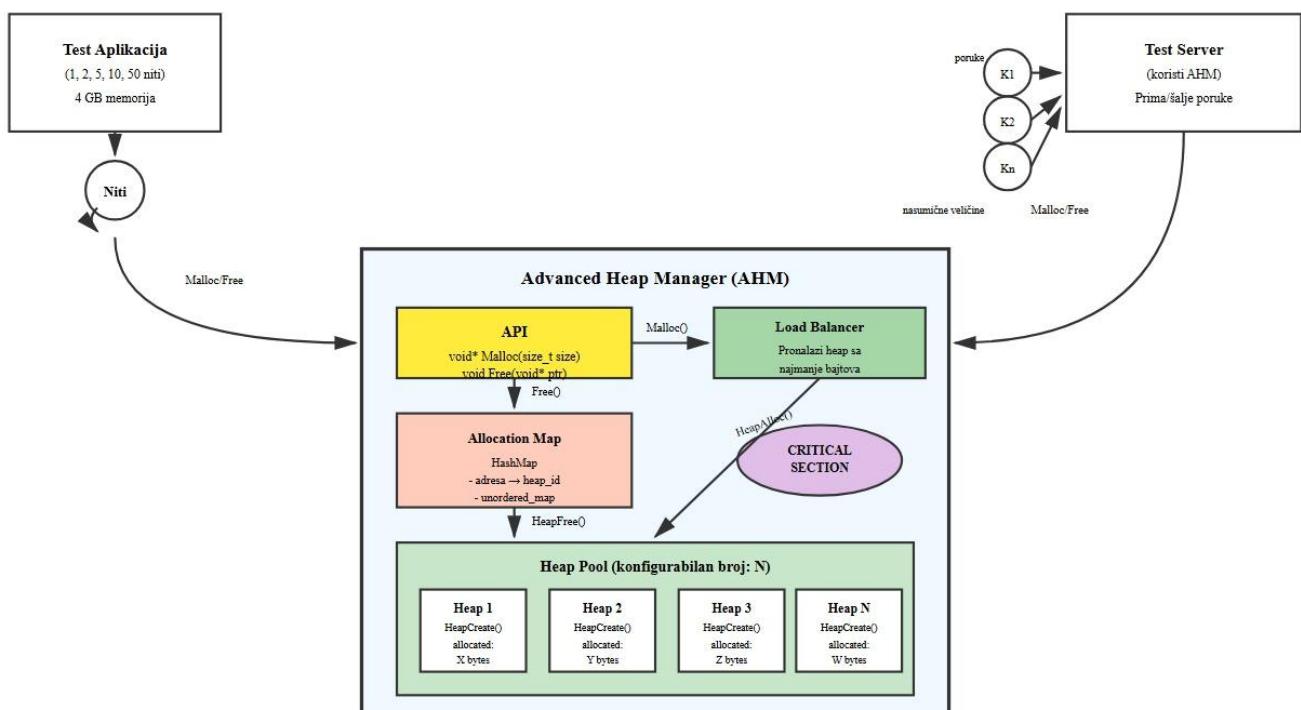
Advanced Heap Manager (AHM) представља напредни систем за управљање меморијом који решава проблем heap connection-а коришћењем више heap инстанци. Уместо да све нити користе један заједнички heap, AHM распоређује захтеве за алокацију на више heap-ова, чиме се смањује број конфликтата између нити и повећава паралелизам система.

AHM користи конфигурабилан број heap-ова, што омогућава прилагођавање система различитим сценаријима рада и оптерећењима. Приликом алокације меморије, AHM бира heap који има најмање алоцираних бајтова, чиме се постиже равномерна расподела оптерећења између heap-ова. Приликом деалокације меморије, систем води рачуна да се меморија врати у heap из ког је првобитно алоцирана, чиме се обезбеђује конзистентност система.

Advanced Heap Manager обезбеђује основне функције за управљање меморијом које су компатибилне са стандардним начином рада апликација. Основне функције које систем имплементира су:

- *void Malloc()** – Функција за алокацију меморије која аутоматски бира оптималан heap за алокацију.
- *void Free(void)** – Функција за деалокацију меморије која враћа меморију у одговарајући heap.

2. Дизајн



а) Тест апликација: тестни програм који директно користи АНМ за мерење перформанси. Садржи различите бројеве нити које тестира у укупном меморијском простору од 4 GB који треба алоцирати и деалоцирати.

Она покреће н нити истовремено. Свака нит позива malloc() за рандом величину и free(), користећи меморију. Такође, мери се време које је потребно да се прође кроз све алокације.

Циљ овог тесла ће упоредити перформансе АНМ са стандардним функцијама и видети да ли АНМ смањује конекцију када има више нити.

б) Thread pool: скуп радник нити које рапалелно извршавају алокације и деалокације. Извршавање је континуирано, нити раде у петљи. Свака нит независно позива функције malloc() и free().

в) Тест сервер: сервер апликација која прима поруке од клијената. Интерно користи АНМ за све алокације и комуницира са клијентима тако што прима или шаље поруке. Док је активан, сервер прими поруку насумичне величине, обради поруку и пошаље одговор клијенту.

г) Клијент (1-n): Више клијената истовремено шаље захтеве серверу. Сваки клијент шаље поруке различитих величина, што форсира сервер да прави више бафера. Више клијената симулира оптерећење система, што доприноси више конкурентних алокација на серверу.

д) API: јавни интерфејс – функције које позивају корисници.

Функција void Malloc(size_t size)* алоцира величине байтова меморије, враћа поинтер на алоцирани простор, а ако алокација не успе, враћа null.

Функција void Free(void *ptr)* деалоцира меморију на коју показује и враћа меморију назад у heap. Позива Allocation Map да пронађе из ког heap-а треба ослободити меморију.

ћ) Load Balancer: компонента која одлучује у који heap да смести нову алокацију. Проналази heap са најмање байтова. Спречава да један heap буде преоптерећен и балансира оптерећење између свих доступних heap-ова.

е) Allocation Map: мапа која памти из ког heap-а је сваки поинтер алоциран. Кључ је адреса која је враћена из malloc(), а вредност је id број heap-а.

ж) Critical section: Механизам за thread-safety, обезбеђује да само једна нит може извршавати критичне операције у једном тренутку. Без њега више нити могу одабрати исти heap, при чему настаје конфликт. Користи се код Load Balancer и око ажурирања allocatedBytes после алокације/деалокације.

з) Heap Pool: колекција н независних heap-ова. Сваки heap представља независтан простор меморије креиран функцијом HeapCreate(). Сваки heap добија свој единствени идентификатор.

Више heap-ова се користи ради паралелизма, где различите нити могу алоцирати из више различитих heap-ова истовремено. Смањује се време чекања и боље су перформансе.

3. Циљ тестирања

AHM функционише боље јер постоји паралелно извршавање, а и боља дистрибуција оптерећења.

Упоређује се време извршавања AHM са стандардним malloc(), прати се број операција у секунди, као и како перформансе расту са бројем нити.

AHM показује боље перформансе са већим бројем нити.