Припреме за лабораторијске вежбе из предмета Системска програмска подршка у реалном времену II
- 2017-2018/ Вежба 10 -

Област вежби: *Паралелно програмирање*

ТВВ, ТУТОРИЈАЛ IV

ДОДЕЛА МЕМОРИЈЕ

Садржај

- Проблеми приликом заузимања меморијског простора
- Решења
- Пример

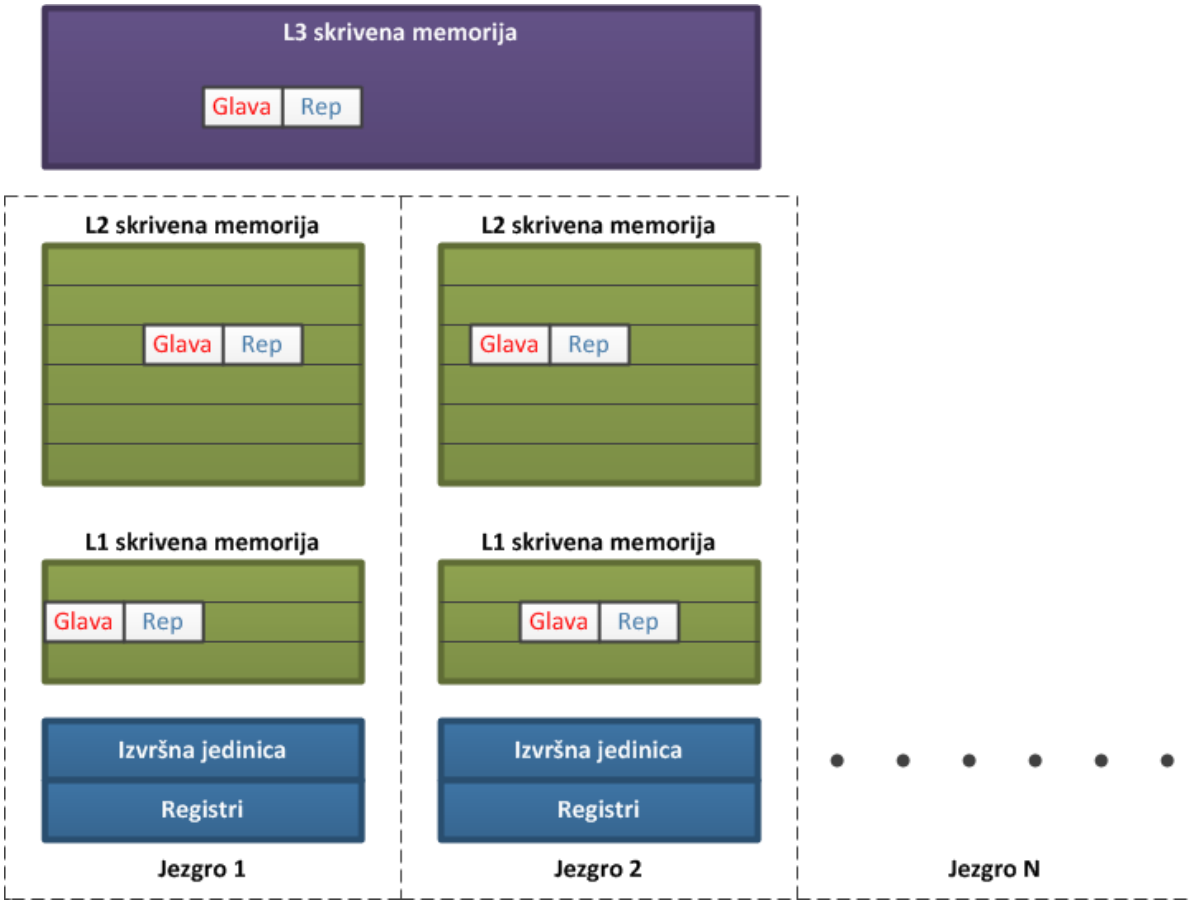
Проблеми приликом заузимања меморијског простора (1/2)

- Секвенцијални програми:
  - једна нит
  - једна динамичка меморија
  - нема проблема ☺
- Паралелни програми
  - више нити (задатака)
  - једна динамичка меморија
  - уско грло: нити се такмиче која ће пре заузети јединствену динамичку меморију тј. део паралелног програма почиње да се извршава секвенцијално ☹

Проблеми приликом заузимања меморијског простора (2/2)

- Проблем лажног дељења
  - Процесори чувају податке из оператривне меморије (RAM) у скривеној меморију (*cache*)
  - Приликом копирање копира се више бајтова – *cache lines* (обично 64 бајта)
  - Јавља се непотребно копирање података из оперативне меморију у скривену меморију ☹
  - Зашто?

Пример лажног дељења



- Језгро 1 пише на локацију ,,Glava”
- Освежава се L2 скривена меморија језгра 1, L3 i RAM меморија
- L1 i L2 скривена меморија језгра 2 више нису валидне
- Језгро 2 чита са локације ,,Rep”
- Вишак (overhead): локације ,,Glava” и ,,Rep” се налазе у истој линији те језгро 2 мора да освежи целу линију пре него што прочита локацију ,,Rep”

Меморијска кашњења

Тип кашњења	Број тактова	Време [ns]
Оперативна меморија		~ 60-80 ns
L3 скривена меморија	~ 40 – 45	~ 20 ns
L2 скривена меморија	~ 10	~ 3 ns
L1 скривена меморија	~ 3-4	~ 1 ns
Регистри	~ 1	

Решења

- Решење 1: scalable_allocator
  - свака нит поседује сопствену динамичку меморију
  - нити се више не такмиче која ће пре добити динамичку меморију, тј. отклања се проблем уског грла ☺
- Решење 2: cache_aligned_allocator
  - структуре се равнају на дужину линије скривене меморије
  - отклања се проблем лажног дељења ☺

scalable_allocator функције

- #include “tbb/scalable_allocator.h”
- void * scalable_malloc(size_t size);
  - заузимање нове меморије
- void * scalable_free(void * ptr);
  - ослобађање меморије
- void * scalable_realloc(void * ptr, size_t size);
  - проширивање или смањивање меморијског простора
- void * scalable_calloc(size_t nobj, size_t size);
  - заузимање меморије за низ елемената

cache_aligned_allocator

- pointer allocate(size_type n)
  - зузимање меморије која је поравната са дужином линије скривене меморије
  - статичка функција класе cache_aligned_allocator<type>()
  - type: тип за који се заузима меморија
- void deallocate(pointer p, size_type n)
  - ослобађање меморије заузете са функцијом allocate
  - статичка функција класе cache_aligned_allocator<type>()
  - type: тип за који се заузима меморија

Напомена!

- Приликом ослобађања заузетог меморијског простора користити функције одговарајућег типа
  - malloc -> free
  - new -> delete
  - scalable_malloc -> scalable_free
  - allocate -> deallocate
- У супротном је могуће неочекивано извршавање програма

Пример