Requirements: Dynamic Memory Management

Alfons Crespo

Memory Management

- There is a significant work done in quality of service of processor scheduling and networks. But memory is a resource that has been included in the list of resources but .... .
- In real-time community there is some myths about dynamic memory management

Myths

1. Allocating dynamic memory (**malloc** operation) is **unbounded and slow**

2. Long running programs will fragment the memory pool more and more, consuming unbounded memory.

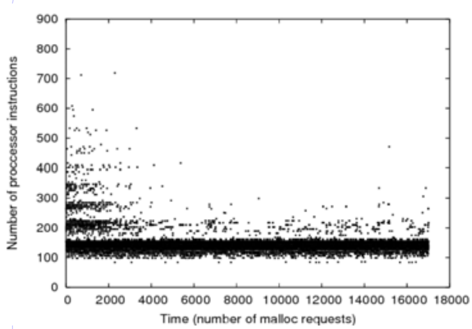
3. It is better to implement your own allocator for your application needs

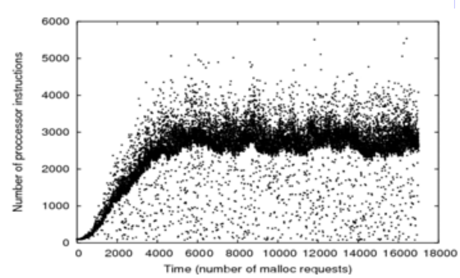
4. Real-time applications should not use dynamic memory

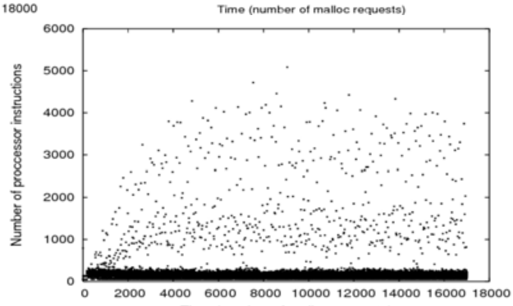
5. ......

Grupo de sistemas de tiempo real. http:/www.gii.upv.es

Allocators







Memory Management

- Current applications (multimedia, mobile, etc.) can require the use of **dynamic memory**.

- Current dynamic storage allocation techniques presents
  - Constant temporal cost (O(1)) for allocating and dealllocating objects (chunks of memory)
  - Low fragmentation (< 15%) in application running for long periods

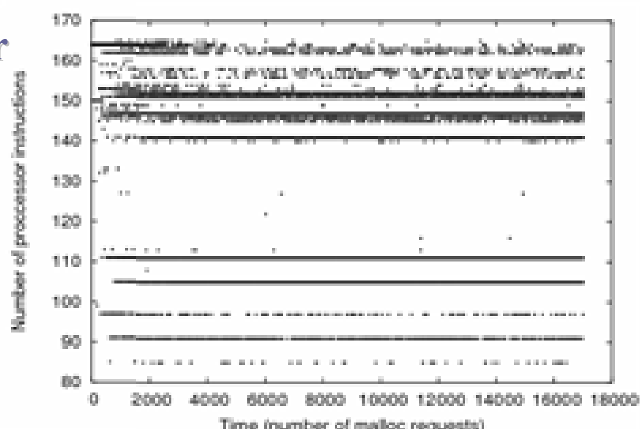
- These features permit to include the temporal cost in the schedulability analysis.

..... but the fragmentation require additional features.

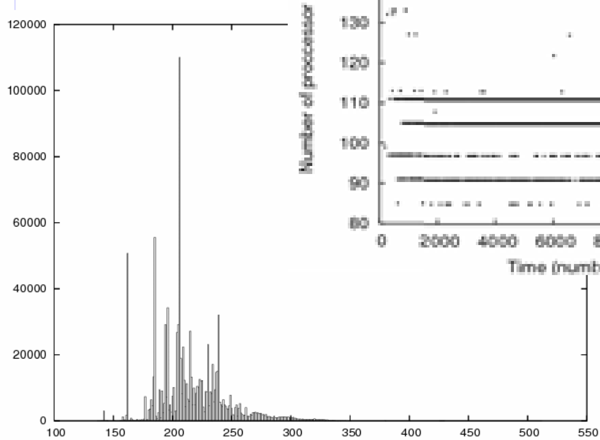
Temporal analysis

TLSF allocator

(OCERA)



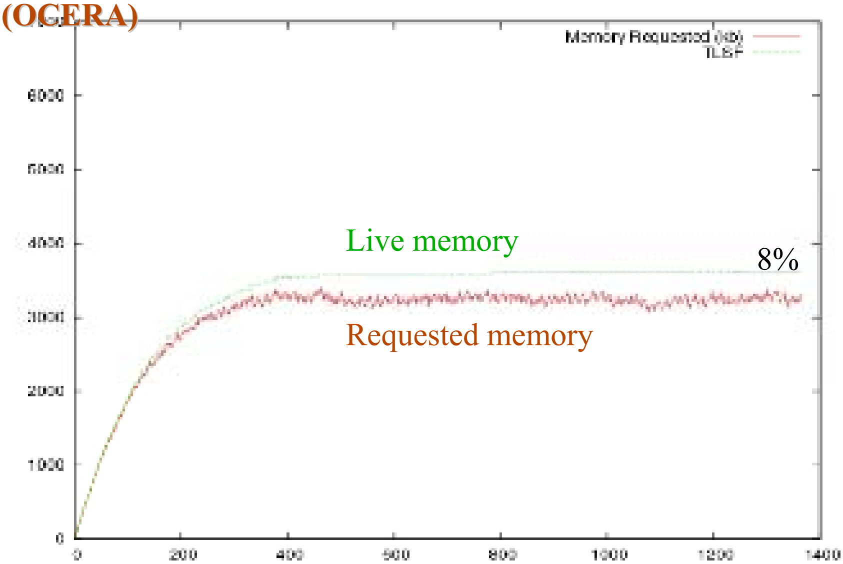
Number of instructions



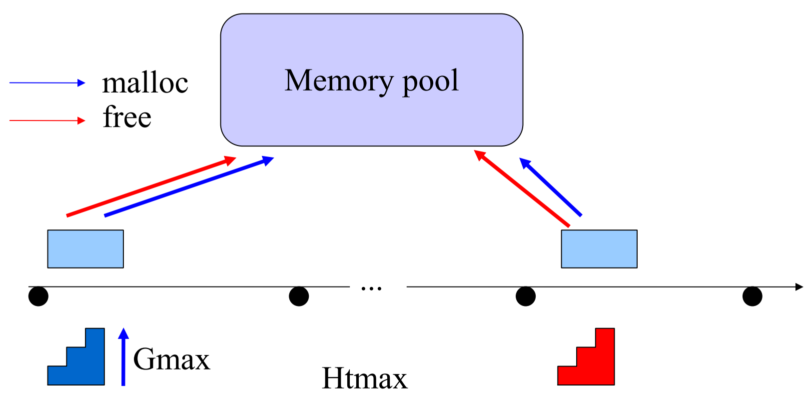
Number of processor cycles histogram

Fragmentation

TLSF allocator
(OCERA)



Real-time task using memory



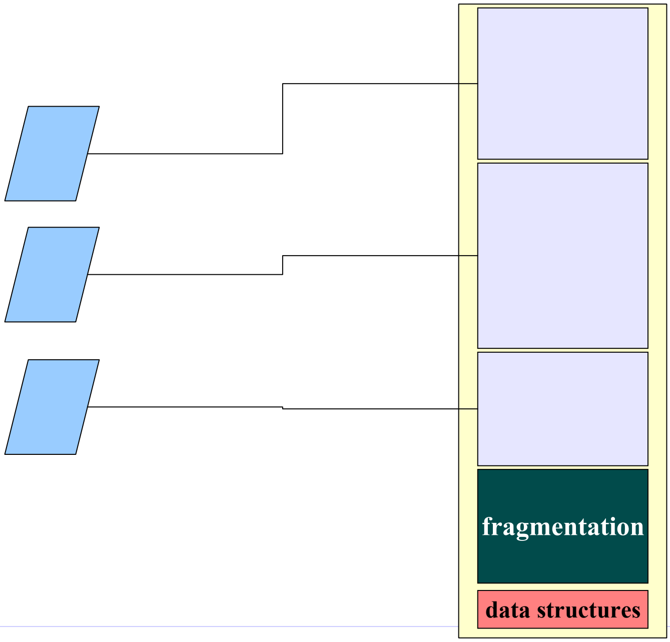
Attributes:

Gmax - maximum amount of memory allocated by period
Htmax - maximum holding time

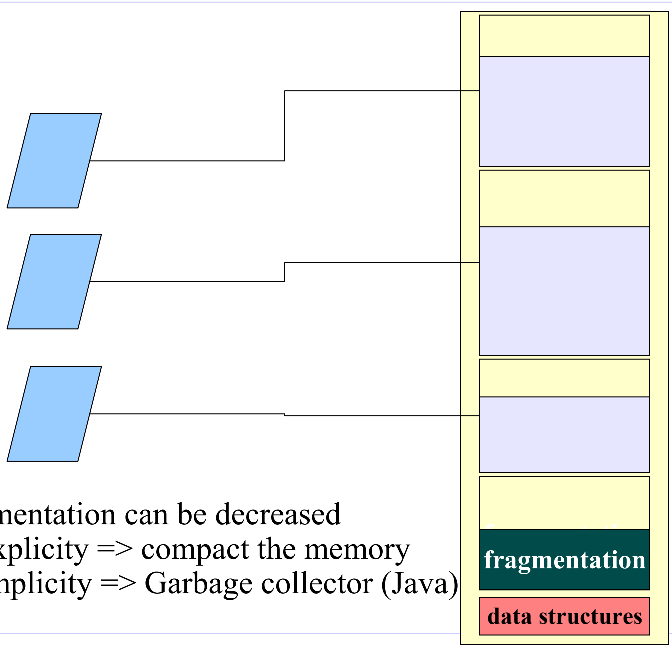
(Ci, Pi, Di, **Gi, Hi**)

Worst Case Memory Space (WCMS) = Gmax *(Htmax / Period)

Real-time task using memory



Real-time task using memory



Fragmentation can be decreased
- explicity => compact the memory
- implicity => Garbage collector (Java)

Real-time task using memory

Memory can be allocated ***permanent*** to a task which can it and free at any moment.

Memory can be allocated ***temporaly*** to a task which can it and free under some constraints (spatial or temporal constrains).

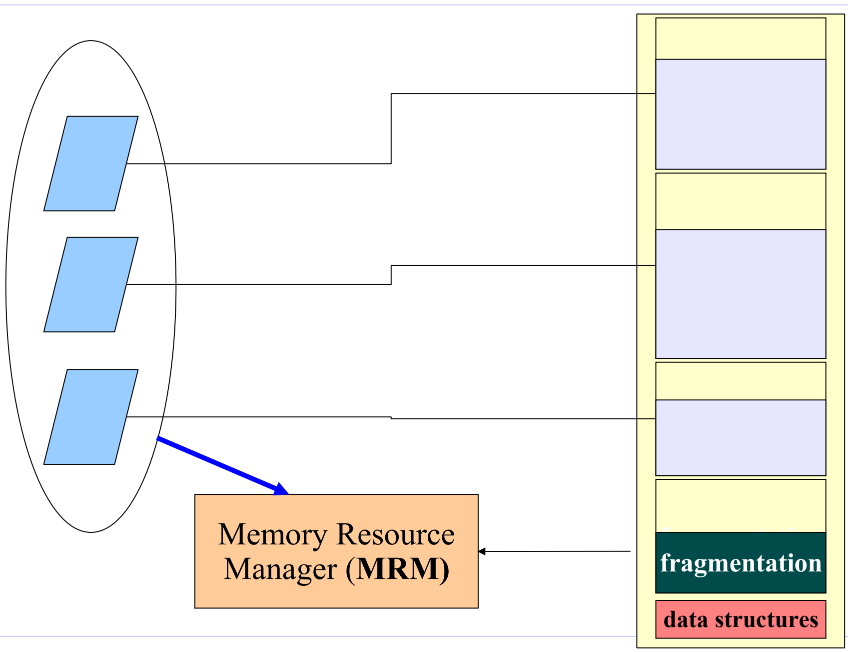
A memory allocation is **permanent** if

Mallocated < Mmax

When no more **permanent** memory can be allocated, more memory can be allocated in a **temporal** way.

- Temporal constraints: during an interval
- Storage constraints: while the memory use is within a range

MRM



MRM Interface

Specify the memory requirements:

MRM_preallocation_parameters(task_id : in Task_Identification,
preallocated_memory_size: in Size,
max_request_period: input Time)

Allocate objects in a permanent way

function **malloc**(task_id : in Task_Identification, size : in Size)
return pointer;

Deallocate objects

procedure **free**(task_id : in Task_Identification, ptr : in Pointer);

MRM Interface

Grupo de sistemas de tiempo real. http:/www.gii.upv.es

Negotiate **new** memory requirements:

- based on temporal requirements

  # A task can ask for more memory (s blocks) for an interval
  # A task can ask for the maximum memory available during k periods

- based on storage requirements

  # A task ask for memory based on storage criteria (not temporal guarantee). Task has to release memory by request.

Allocate objects in a temporal way

tmalloc

MRM

Also, other kind of tasks can be considered (Non periodic tasks)

- run for a number of periods (under server constraints)

or / and

- have global memory needs (maximum memory by applications)