TLSF: Timing and spatial study

Miguel Masmano, Ismael Ripoll y Alfons Crespo
Dept. Informática de Sistemas y Computadores (DISCA)
Universidad Politécnica de Valencia (Spain)

Outlines

- Introduction
- Overview of the TLSF
- Timing evaluation
- Spatial evaluation (fragmentation)

Introduction

- The usage of Dynamic Storage Allocation (DSA) in Real-Time Systems (RTS) has been avoided due to two reasons:
  - Either a non-deterministic timing response or a unsuitable one
  - The **fragmentation** problem
- However, DSA is more and more neccesary:
  - More and more complex applications
  - Usage of programming languages with DSA requirements: Java
  - Memory-restricted systems
- Explicit DSA
  - No garbage collection

TLSF: a DSA manager with a constant response time

- A good-fit policy is implemented through a set of segregated lists where blocks of similar sizes are grouped
- TLSF does not look for free blocks within its structure but non-empty lists
- Via mapping functions and bitmaps a suitable non-empty list is found without using either iteration or recursivity
- Constant response time!

TLSF

i \ j	0	1	2	3	4	5	6	7	mapa_bits_j[]
31	$2^{31}+0*2^{28}$	$2^{31}+1*2^{28}$	$2^{31}+2*2^{28}$	$2^{31}+3*2^{28}$	$2^{31}+4*2^{28}$	$2^{31}+5*2^{28}$	$2^{31}+6*2^{28}$	$2^{31}+7*2^{28}$	00000000
:	. . . . . .		. . . .			. . .			
16	$2^{16}+0*2^{13}$	$2^{16}+1*2^{13}$	$2^{16}+2*2^{13}$	$2^{16}+3*2^{13}$	$2^{16}+4*2^{13}$	$2^{16}+5*2^{13}$	$2^{16}+6*2^{13}$	$2^{16}+7*2^{13}$	00100000
:	. . .	. . .	. . . .		. . .				
7	128	144	160	176	192	208	224	2^7+7*2^4	00000000
6	64	72	80	88	96	104	112	120	00000100
5	32	36	40	44	48	52	56	60	01000000

libre
libre
libre
libre
libre
libre
libre

110...1...0

mapa_bits_i

i	12	16	20	24	28	10101

libre
libre
libre
libre
libre

Comparative timing study

1) Selection among several allocators
   - Real-Time suitability (TLSF, Half-fit,Bin-buddy, AVL)
   - Popularity (DLmalloc)
   - As reference (first/best fit)
2) Studied the asymptotic cost of each considered allocator
3) Design of the worst/bad timing scenario for each allocator:
   - Number of instructions executed
   - Processor Cycles (TSC)

Selected DSA allocators

- First-Fit: First-fit policy implemented by a doubly linked list
- Best-Fit y AVL malloc: Best-fit policy implemented via a doubly linked list and a AVL tree respectively respectivamente
- Binary Buddy: “Buddy” policy implemented through segregated lists
- Dlmalloc: Exact-fit and best-fit policies implemented by segregated lists and a doubly indexed list
- TLSF y Half-Fit: Good-fit policy implemented by segregated lists

Asymptotic timing analysis

- H: Heap size
- n: Smaller block size

	Allocation	Deallocation
First-fit/Best-fit	$O\left(\mathcal{H}/\left(2\cdot n\right)\right)$	$O(1)$
Binary-buddy	$O\left(\log_2\left(\mathcal{H}/n\right)\right)$	$O\left(\log_2\left(\mathcal{H}/n\right)\right)$
DLmalloc	$O\left(\mathcal{H}/n\right)$	$O(1)$
AVL-tree	$O\left(2\cdot 1{,}44\cdot\log_2\left(\mathcal{H}/n\right)\right)$	$O\left(3\cdot 1{,}44\cdot\log_2\left(\mathcal{H}/n\right)\right)$
Half-fit/TLSF	$O(1)$	$O(1)$

Allocation results: Worst Scenario

Malloc	FF	BF	BB	DL	AVL	HF	TLSF
FF WC	**81995**	**98385**	115	109	699	162	197
BB WC	86	94	**1403**	729	353	162	188
DL BC	88	96	1113	**721108**	353	164	197
AVL BC	5085	6093	252	56093	**3116**	162	197
TLSF WC	88	96	1287	729	3053	**164**	**197**

Processor Instructions

Malloc	FF	BF	BB	DL	AVL	HF	TLSF
FF WC	**161326**	**158755**	1445	1830	6471	1633	2231
BB WC	1168	1073	**3898**	4070	3580	1425	2388
DL BC	1203	1227	3208	**331325**	3844	1651	2251
AVL BC	105835	101497	1703	13216	**11739**	1629	2149
TLSF WC	1168	1074	3730	4124	3580	**1690**	**2448**

Processor Cycles

H: Heap size = 8Mbytes
n: Smaller block size = 16 bytes

Deallocation results: Worst Scenario

Free	FF	BF	BB	DL	AVL	HF	TLSF
Proc. Instr.	115	115	1379	51	1496	130	187
Proc. Cycles	1241	1289	4774	955	7947	1110	2151

Maximun theoretical fragmentation

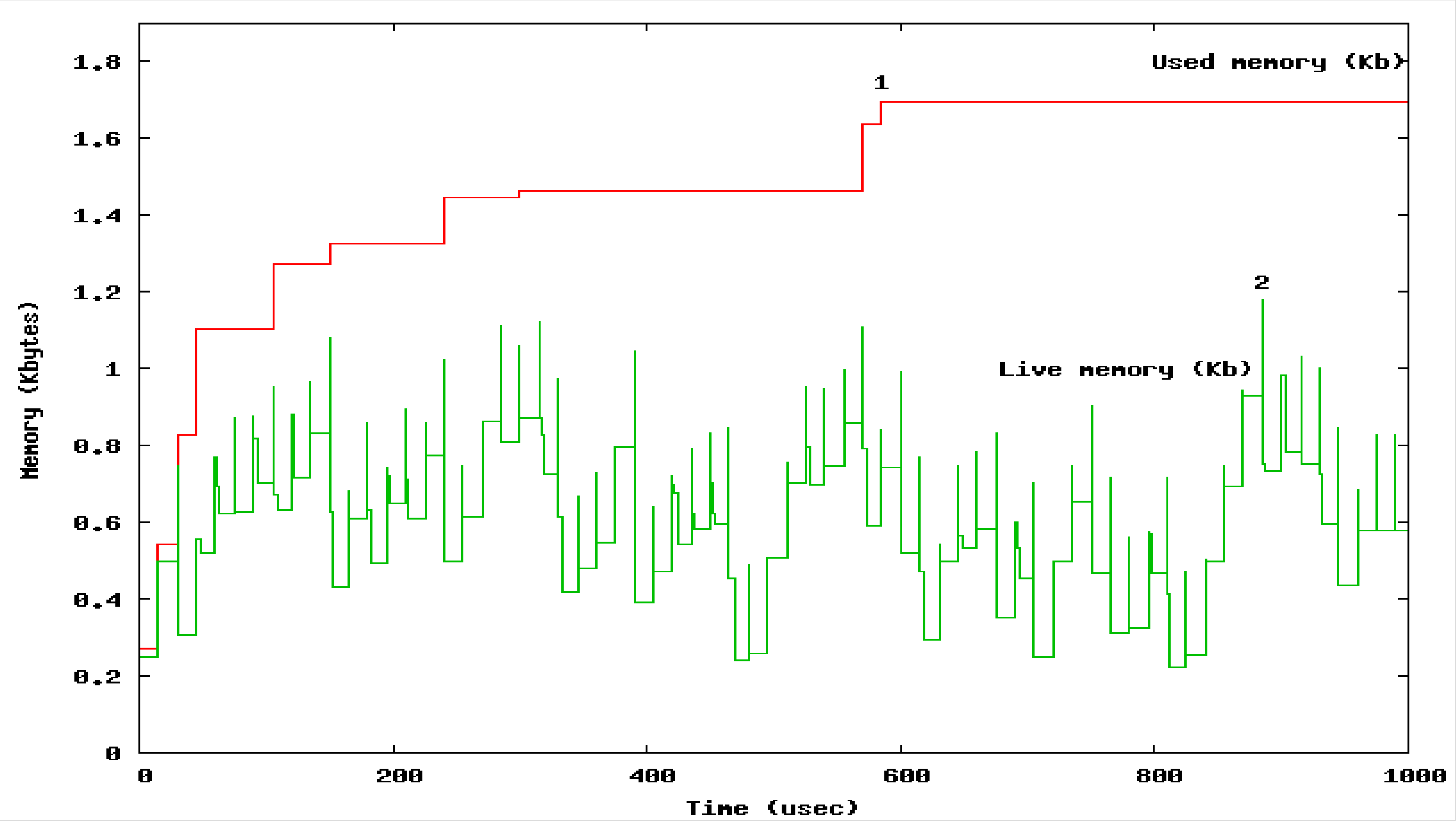
- M: Maximun live memory
- m: Size of the larger required block

  - First-Fit: $M*(1+\log_2(m))$
  - Best-Fit y AVL: $O(M*m)$
  - Binary Buddy: $2*M*\log_2(m)$
  - DLmalloc: $O(M*m)$
  - TLSF y Half-Fit: $O(M*m)$

The worst cases are amazingly pessimistics !

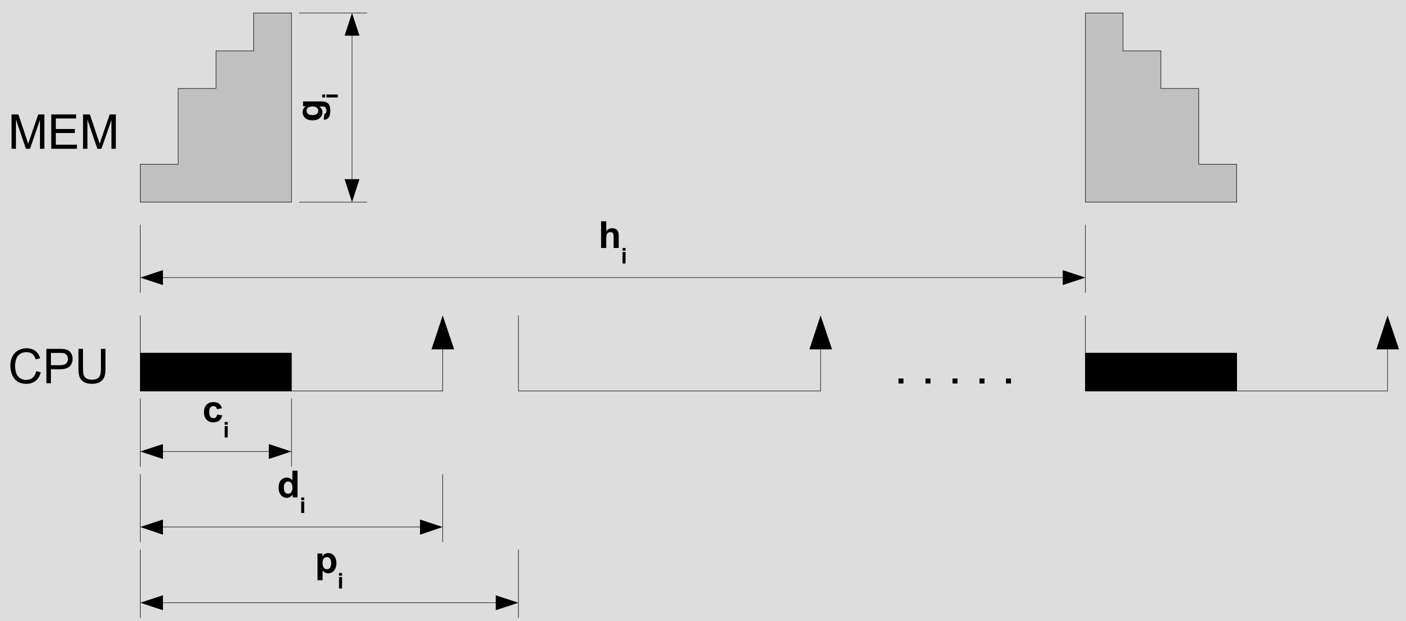
Fragmentation metric

- Fragmentation = Point of maximun memory usage of the allocator relative to the point of maximun live memory. For example, in the figure: Fragmentation = 47.06 %



Periodic task model with dynamic memory

- Periodic task definition T_i: $T_i=(p_i, c_i, d_i, \mathbf{g_i}, \mathbf{h_i})$
  - $\mathbf{g_i}$: Maximun amount of memory required per activation
  - $\mathbf{h_i}$: Maximun number of periods that a memory block can stay allocated



Fragmentation results from the simulation

Gestor		FF	BF	BB	DL	AVL	HF	TLSF
C1 (%)	Mean	100.01	4.64	46.37	5.19	4.64	82.23	**4.33**
	Typ. Desv.	4.96	0.61	0.74	1.03	0.61	1.06	**0.55**
C2 (%)	Mean	85.01	**4.54**	45.00	6.09	**4.54**	75.15	4.99
	Typ. Desv.	4.92	0.67	0.98	0.92	0.67	1.52	**0.59**
C3 (%)	Media	112.51	**7.01**	48.63	10.43	**7.01**	99.10	7.69
	Desv.Tip.	8.53	1.13	1.90	1.54	1.13	2.61	**0.98**
C4 (%)	Mean	109.71	22.09	69.59	40.63	22.09	73.58	**12.52**
	Typ. Desv.	14.04	1.84	15.13	10.33	1.84	5.55	**1.57**

Conclusions

- The most part of existing allocators have been designed keeping in mind the average response time
  - Unsuitable for RTS
- Before Half-Fit and TLSF were created, it was widely belived that a low response time meant high quantity of usage of memory (high fragmentation)

Conclusions

- *A priori,* TLSF, AVL malloc, Half-Fit and Binary Buddy seem to be suitable for RTS
  - All of them show a quick, deterministic response time
  - TLSF and AVL malloc show a low fragmentation
  - Half-Fit and Binary Buddy show a high fragmentation (compared with the rest)
- Therefore, TLSF and AVL malloc show quite good features to be used in RTS

Conclusions (Summary)

	Timing	Spatial
First-Fit	✘	✔
Best-Fit	✘	✔✔
AVL	✔	✔✔ O(log(H))
Binary-Buddy	✔	✘✘
Dlmalloc	✘	✔
Half-Fit	✔✔	✘✘
TLSF	✔✔	✔✔ O(1)