Dynamic memory allocator implementations in Linux system libraries

Wolfram Gloger
Poliklinik für Zahnerhaltung
Goethestr. 70
80336 München
wmglo@dent.med.uni-muenchen.de

1. Overview

2. Introduction

Good dynamic memory allocation is essential for most computer applications:

- Static allocation is fast, but inflexible. Fixed limits are **bad.**
- Simple allocation disciplines (e.g. LIFO/stack) are rarely enforcible.
- Memory (RAM) is almost always scarce.

Ref. [Wilson95] estimated the worldwide cost of the widepread use of poor allocators (main & cache memory, CPU cycles) upwards of a billion U.S. dollars.

Avoidance of system library allocators in favor of **ad hoc** solutions is still common, because of extreme concerns about speed and memory costs of general heap allocation.

3. Chunk layout in DL-malloc

Every chunk is aligned on an 8-byte address, with **boundary tags**:

Previous chunk	...	
Start of chunk	prev_size	
	size|flags	
	forw. link	Start of usable area

	back link	
	...	
Next chunk	prev_size	End of usable area

Flags stored in least significant bits of the size field:

- PREV_INUSE (previous chunk in use)
- IS_MMAPPED (chunk is mmapped)

Features:

- Chunks can be split and coalesced arbitrarily.
- Fixed overhead is `sizeof(size)` (normally 4 bytes) per chunk.
- Minimum chunk size is typically 16 or 24 bytes.

4. Malloc tracing facility

(available at `ftp://ftp.dent.med.uni-muenchen.de/pub/wmglo/mtrace.tar.gz`) Analysis of real-world allocation patterns in two steps:

1. Record all use of malloc(), free(), realloc() and sbrk() by an application in a file (by pre-loading a special shared library, malloc-trace.so).
2. Replay the recorded allocation pattern through a simulator (trace-test), linked against different malloc implementations.

5. Malloc implementations tested in these experiments

Name	Author	Version	Source
vmalloc	Phong Vo		`www.att.com`
Emacs	M. Haertel et.al.	19.34	GNU Emacs 19.34
FreeBSD	P.-H. Kamp	1.18	FreeBSD-current
CSRI	Mark Moraes	1.18	`ftp.csri.toronto.edu`
BSD	Ch. Kingsley et.al.		perl-4.036
GC	H.-J. Boehm	4.11	`reality.sgi.com`
Linux/DL	Doug Lea	2.6.4	libc-5.4.23
SGI			Irix 5.3

6. Definition of performance criteria

For a given trace-file, the following variables are evaluated:

- User time
- System time
- Peak allocation (maximum of all allocated heap memory, largest footprint)
- Waste at peak allocation time (very important)

- Average waste

The `Waste rate', or `fragmentation', is defined as:

$$f = 1 - \frac{\text{Number of bytes allocated by application}}{\text{Number of bytes used from system}}$$

Worst case lower bound for f:

$$f \geq \text{const} \cdot \log_2 \frac{\text{largest object size}}{\text{smallest object size}}$$

7. Random allocation patterns with uniform distribution

5,000,000 random allocations/frees/reallocations with maximum chunk size of 1,000.

Implement.	User /sec	Sys /sec	Waste/%
vmalloc	50.2	0.41	15.3
Emacs	34.0	0.34	31.3
FreeBSD	37.3	0.49	30.3
CSRI	141	0.28	16.7
GC	202	0.56	76.0
Linux/DL	40.5	0.37	9.46
SGI	52.3	0.36	16.6

8. Preprocessing for C

(average over all runs of `cpp` from gcc-2.7.2 when making ddd-2.0)

Operations	967690
malloc	659202
free	307889
realloc	599
max. size	1578758
avg. size	517489
time	81.5478sec

Implement.	User	Sys	Peak allocation	Waste at peak	Average waste

	/sec	/sec	/MB	/%	/%
vmalloc	5.67	14.2	2.31	37.5	31.6
Emacs	6.14	14.0	2.33	38.0	37.9
FreeBSD	6.00	13.6	2.29	36.9	35.8
CSRI	5.88	13.7	2.19	33.0	31.6
BSD	4.91	14.5	3.28	59.3	54.2
GC	30.9	17.0	5.99	75.5	73.0
Linux/DL	4.38	13.6	1.74	9.18	18.8
SGI	7.13	16.5	2.16	33.2	28.0

9. Compilation of large C++ sources

(average over all runs of `cc1plus` from gcc-2.7.2 when making ddd-2.0)

Operations	461951
malloc	293606
free	167635
realloc	710
max. size	12780437
avg. size	3432528
time	617.7394sec

Implement.	User /sec	Sys /sec	Peak allocation /MB	Waste at peak /%	Average waste /%
vmalloc	3.70	17.7	12.9	1.19	4.60
Emacs	3.18	10.4	13.0	1.89	4.25
FreeBSD	3.20	10.1	12.9	1.48	2.98
CSRI	3.87	18.8	12.8	0.66	2.24
BSD	2.64	19.3	13.2	14.6	3.12
GC	85.3	19.5	16.8	25.0	38.1
Linux/DL	3.34	18.6	12.8	0.51	1.45
SGI	5.11	22.7	12.8	0.42	1.91

10. X server traces

(average over three different X server sessions - xterms, 3D-visualization application, web-surfing, Emacs, xdvi)

Operations	1650126
malloc	794497
free	782568
realloc	73061
max. size	3655661
avg. size	1322521
time	12678.688sec

Implement.	User /sec	Sys /sec	Peak allocation /MB	Waste at peak /%	Average waste /%
vmalloc	7.19	1.36	6.89	58.2	61.1
Emacs	6.55	1.51	4.98	43.9	40.7
FreeBSD	7.21	1.43	4.42	19.1	39.2
CSRI	8.02	1.43	5.47	48.4	62.0
BSD	5.16	1.45	8.36	69.2	73.6
GC	39.0	6.62	49.0	98.9	91.3
Linux/DL	5.95	1.58	3.79	3.48	12.1
SGI	8.26	1.74	4.74	41.0	52.2

11. Web surfing with Netscape

(a single 20-minute Netscape-3.01 session)

Operations	513001
malloc	263318
free	241833
realloc	7306
max. size	2592344
avg. size	1761078
time	1096.9718sec

Implement.	User /sec	Sys /sec	Peak allocation /MB	Waste at peak /%	Average waste /%
vmalloc	1.89	0.460	3.14	31.2	30.2
Emacs	1.87	0.464	3.29	25.2	29.7

FreeBSD	2.10	0.438	3.11	17.2	26.7
CSRI	1.96	0.464	2.87	24.8	26.3
BSD	1.41	0.472	5.18	53.5	57.8
GC	5.24	0.675	10.5	76.6	79.8
Linux/DL	1.69	0.498	2.83	8.84	19.0
SGI	2.58	0.574	2.91	10.9	27.5

12. Word processing/large GUI application

(1/4-hour swriter3 session; StarOffice-3.1beta4)

Operations	915638
malloc	460902
free	448060
realloc	6676
max. size	3794066
avg. size	2197864
time	499.1156sec

Implement.	User /sec	Sys /sec	Peak allocation /MB	Waste at peak /%	Average waste /%
vmalloc	3.61	0.849	4.83	21.5	24.3
Emacs	3.50	0.844	4.80	20.9	25.8
FreeBSD	3.90	0.740	4.88	22.3	28.8
CSRI	3.85	0.748	4.68	18.9	21.1
BSD	2.65	0.808	7.48	67.9	48.3
GC	6.69	1.08	15.6	84.5	75.1
Linux/DL	3.25	0.750	4.06	6.46	10.8
SGI	4.24	0.878	4.48	15.3	21.6

13. Special features of the malloc in Linux libc

- Large chunks are allocated via `mmap()`. The threshold is set through the environment variable MALLOC_MMAP_THRESHOLD_. The maximum number of chunks allocated in this way is limited through MALLOC_MMAP_MAX_.
- Trimming occurs when the size of the top chunk exceeds MALLOC_TRIM_THRESHOLD_.
- Whenever `sbrk()` is called to obtain more memory, an extra MALLOC_TOP_PAD_

bytes is allocated (hysteresis).
- Debugging simple usage errors can be turned on at runtime by exporting MALLOC_CHECK_ (0: ignore errors silently; 1: print error message and exit; 2: abort)

14. ptmalloc - a fast multi-thread version of Doug Lea's malloc

(available with glibc-2.x and also separately at `ftp://ftp.dent.med.uni-muenchen.de/pub/wmglo/ptmalloc.tar.gz`) Multiple arenas, individually lockable:

TSD thread 1	TSD thread 2	...
Arena1	**Arena 2**	...
Mutex	Mutex	...
Bins	Bins	...
Heap 1a	**Heap 2a**	...
aArena1	aArena2	...
Chunks...	Chunks...	...
Heap 1b	**Heap 2b**	...
aArena1	aArena2	...
Chunks...	Chunks...	...
...	...	...

All `heaps' are aligned on a large-power-of-two boundary, for easy and fast chunkaarena mapping. The maximum heap size is much larger than the mmap threshold. Heaps are allocated with `mmap()` and can grow and shrink individually.

Threads lock the first arena they can get for all allocation operations; the one used last is remembered in TSD.

15. Performance results from ptmalloc

Sun Sparc, two TI,TMS390Z55 50MHz CPUs, Solaris 2.4:

40 threads (2 in parallel), 10,000,000 malloc calls per thread, max. size 25,000

time	ptmalloc	libc-malloc
real	1:18:54.0	3:10:50.8
user	2:09:33.0	3:08:39.5
sys	20:20.3	4:02.9

SGI Indy, one R4600 100MHz CPU, Irix 5.3:

20 threads (4 in parallel), 500,000 malloc calls per thread, max. size 10000

time	ptmalloc	libc-malloc
real	1:49.15	59:52.10
user	1:36.30	15:59.54
sys	4.83	36:19.53

16. References

1. Doug Lea: **A Memory Allocator**. **unix/Mail** December, 1996. Hanser Verlag. (http://g.oswego.edu/dl/html/malloc.html)
2. Paul R. Wilson, Mark S. Johnstone, Michael Neely and David Boles: **Dynamic Storage Allocation: A Survey and Critical Review**. Proc. 1995 Int'l Workshop on Memory Management. Springer LNCS. (ftp://ftp.cs.utexas.edu/pub/garbage/allocsrv.ps)
3. Benjamin Zorn and Dirk Grunwald: **Empirical measurements of six allocation-intensive C programs.** Tech. Rep. CU-CS-604-92, University of Colorado at Boulder, Dept. of CS, July 1992.

File converted from TEX by tth, version 0.6.