

Memory Consumption Analysis with liballoc/Heapchecker



Applies to:

Application Server kernel, C/C++ development and support. For more information, visit the [Performance homepage](#).

Summary

SAP liballoc/Heapchecker is a memory analysis tool for C/C++ programs. It runs on 64 bit Linux, Solaris, HP-UX and as a prototype on Windows. *Heapchecker* is a fast and light weight solution, what makes its usage possible for development phase at SAP and also for troubleshooting on customer side. It helps in finding memory leaks, buffer overwrites and memory management anti-patterns.

Author: G. Maly

Company: SAP

Created on: 1 October 2010

Author Bio

Gennady Maly is involved in C/C++ development at SAP since 2001.

Table of Contents

Why Heapchecker?	3
Functionality	3
Memory Statistics Output.....	4
Memory Corruption Output	6
Conclusion	7
Related Content	8
Copyright	9

Why Heapchecker?

Well known situation: an executable on the server consumes a lot of main memory and nobody knows why. It get worse if support and development teams can't reproduce the issue on a test system. The question is, how can we localize the issue and identify the functions, classes or at least libraries which are responsible for high memory consumption?

We concentrate us here at this kind of memory management problems in C/C++ programs written at SAP, where are e.g. :

- *disp+work, message server, enqueue, ICM* – parts of ABAP AS
- *jstart* – Java AS launcher,
- *TREX IndexServer, NameServer, RFC Server* – parts of TREX engine
- and many other...

We discuss in this article dynamic memory allocation via malloc/new only. Shared memory consumption is not a topic of this article. Shared memory used by ABAP runtime can be checked with other SAP tools like [ABAP Memory Inspector](#).

SAP Java AS memory can be explored with e.g. [Java Memory Analyzer](#).

Functionality

Heapchecker does use the dynamic libraries preload mechanisms of the operating system. On the most of UNIX systems it can be done with an LD_PRELOAD environment variable setting to bring dynamic loader to bind a running executable against the user defined symbols. Example for a usage in a bash shell:

```
> export LD_PRELOAD=./liballoc.so
> sappfpar pf=SampleProfile
```

liballoc.so is the name of Heapchecker library and *sappfpar* is a simple SAP application to be analyzed here.

The following four functions are defined in liballoc/Heapchecker library for overloading standard libc memory management:

```
void* malloc (size_t size);
void free (void * ptr);
void* calloc (size_t n, size_t sz);
void* realloc (void * old_ptr, size_t sz);
```

It is enough for (SAP) applications written in C and C++ as well as new/delete will be implemented internally via malloc/free by all the relevant compilers.

This way of function overloading does make “bookkeeping” of heap allocations possible to Heapchecker. It protocols the timestamp, size and other information about every allocation/deallocation and performs requested allocation/deallocation afterwards, so that it remains transparent for the application being analyzed. Recording will be done in main memory in a thread safe manner, reports will be written into log files if required.

Memory Statistics Output

Here are several log file fragments.

List of chunks sizes in process pid 16399. Time 22:54:49.160

Size	Chunks	KB	LowPeak	HighPeak	KB	Incr	AllocCount
1	3	0	3	4	0	1	6012
2	22	0	22	24	0	1	731
3	31	0	31	34	0	1	3330
4	697	2	697	37269	145	8	16179244
5	38	0	38	42	0	1	9744
6	53	0	53	56	0	1	6250
7	17	0	17	21	0	1	5471
8	668521	5222	668521	1438761	11240	3	8306936
9	22	0	22	29	0	1	7680
10	11	0	11	18	0	1	1167
11	14	0	14	16	0	1	463
12	100	1	100	478	5	7	699408
13	2277	28	2277	4585	58	6	6614395
14	16185	221	16185	18375	251	0	18424016
15	6858	100	6858	7457	109	5	5225828
16	2955	46	2955	143544	2242	8	10367315
...							
...							
66767088	1	65202	1	1	65202	0	2
67904000	2	132625	0	2	132625	0	2
...							
...							

Total: 55117 sizes, 2805961 chunks 6781 MB, 305 M allocs

The snippet above is a distribution statistic about allocation sizes of a running process. We can see e.g. that there are currently only 3 chunks (allocations) of size 1 Byte and exactly 668521 8-Byte-chunks. The number of 8-Byte-chunks have been even higher during the applications runtime: 1438761 was the peak value.

The largest chunk size is 67904000 Byte. There are 2805961 chunks allocated, they occupy 6781 MB net on the heap. This kind of information is useful then analyzing memory fragmentation issues.

The next snippet is an example of a memory leak suspicion report. Heapchecker concludes that the number of chunks with size 133104 has grown continuously. There are 44 of them currently at 19:30:40. This fact will be reported inclusive the call stack of the questionable allocation.

```
19:3:40.144 Leak state of size 133104: chunks 44 (lo 44, hi 46, incr 0), allocs 126
Queue:      35      36      37      38      39      40      41      42      43      44
           18:15:16 18:21:4. 18:23:46 18:28:38 18:29:14 18:32:29 18:33:18 18:38:59 18:42:33 19:3:40
```

Chunks all sizes: 2777809 6611.608 MB, Chunks of size 133104: 45 5.729 MB

19:3:40.144 Number of allocated chunks per size

```
-----+-----
up to |      1      2      4      8     16     32     64    128    256    512
-----+-----
Byte |      3     22   1011  663796  28199  60825   7802  255779 134107 305367
KB   | 1078412 220327  1719   2182   2407   3120   3511   3176   2261  1494
MB   |   1076    720    279    200     7     3     2     2     0     0
GB   |     0     0     0     0     0     0     0     0     0     0
```

19:3:40.144 *** StackTrace ***

```
[Thr 1109473600] ----- C-STACK -----
/usr/sap/BAA/TRX00/exe/liballoc_mt.so[0x2b17bb96758e]
/usr/sap/BAA/TRX00/exe/liballoc_mt.so(_Z10CTrcStack2P8_IO_FILEPv+0x5f) [0x2b17bb9679cb]
/usr/sap/BAA/TRX00/exe/liballoc_mt.so(_Z9CTrcStackP8_IO_FILE+0x1a) [0x2b17bb967a26]
/usr/sap/BAA/TRX00/exe/liballoc_mt.so(_Z10StackTraceP8_IO_FILE+0x15) [0x2b17bb95fd5]
/usr/sap/BAA/TRX00/exe/liballoc_mt.so[0x2b17bb951f3f]
/usr/sap/BAA/TRX00/exe/liballoc_mt.so[0x2b17bb952321]
/usr/sap/BAA/TRX00/exe/liballoc_mt.so[0x2b17bb95285c]
/usr/sap/BAA/TRX00/exe/liballoc_mt.so(_Z19MY_MALLOC_HEAPCHECKm+0x1645) [0x2b17bb953ed7]
/usr/sap/BAA/TRX00/exe/liballoc_mt.so(malloc+0x100) [0x2b17bb94bd38]
/usr/lib64/libstdc++.so.6(_Znwm+0x1d) [0x2b17c459dfcd]
/usr/sap/BAA/TRX00/exe/ims_search_api.so(_ZN8Executor11BwPopSearch10executePopERKN4_STL6vectorIPNS_8PlanDataENS1_9allocatorIS4_EEEES9_RKNS_13ExecutionInfoE+0xd85) [0x2b17bdfadea3]
/usr/sap/BAA/TRX00/exe/executor.so(_ZN8Executor13PlanOperation10executePopERKN4_STL6vectorIPNS_8PlanDataENS1_9allocatorIS4_EEEES9_RN17TRexCommonObjects12TRexApiErrorERKNS_13ExecutionInfoE+0x16) [0x2b17bd55977e]
/usr/sap/BAA/TRX00/exe/executor.so(_ZN8Executor2X210runPopTaskERNS0_11PopTaskInfoERi+0x97c) [0x2b17bd58825c]
/usr/sap/BAA/TRX00/exe/executor.so(_ZN8Executor2X212runPopThreadERNS0_11PopTaskInfoE+0xab) [0x2b17bd589065]
/usr/sap/BAA/TRX00/exe/executor.so(_ZN8Executor8X2Thread3runEPv+0xa4) [0x2b17bd58a46c]
/usr/sap/BAA/TRX00/exe/TrexBase.so(_ZN11TrexThreads10PoolThread3runEv+0x976) [0x2b17bf706f3a]
/usr/sap/BAA/TRX00/exe/TrexBase.so[0x2b17bf708187]
/lib64/libpthread.so.0[0x2b17c494a143]
[Thr 1109473600] -----
```

Such a pattern – growing number of allocations of determined size – is often an indicator for a memory leak. Heapchecker is also able to revert its memory leak suspicion if the number of allocations of the determined size get reduced.

Memory Corruption Output

Another operation area of Heapchecker are heap corruption issues like buffer overflow. Here a log example:

```
pid 11548 check malloc damaged adr      0x6608d90 size 100
eye_before      0x6608d88 MALL>>>> 4D 41 4C 4C 3E 3E 3E 3E
eye_after       0x6608df4 w<<<MALL 77 3C 3C 3C 4D 41 4C 4C
malloc chain error detected
```

One of the challenges in buffer overflow finding is that an overflow may not cause a runtime error immediately, but some time later, maybe in another function, thread etc. The goal of the Heapchecker here is to assert an overflow immediately after it took place.

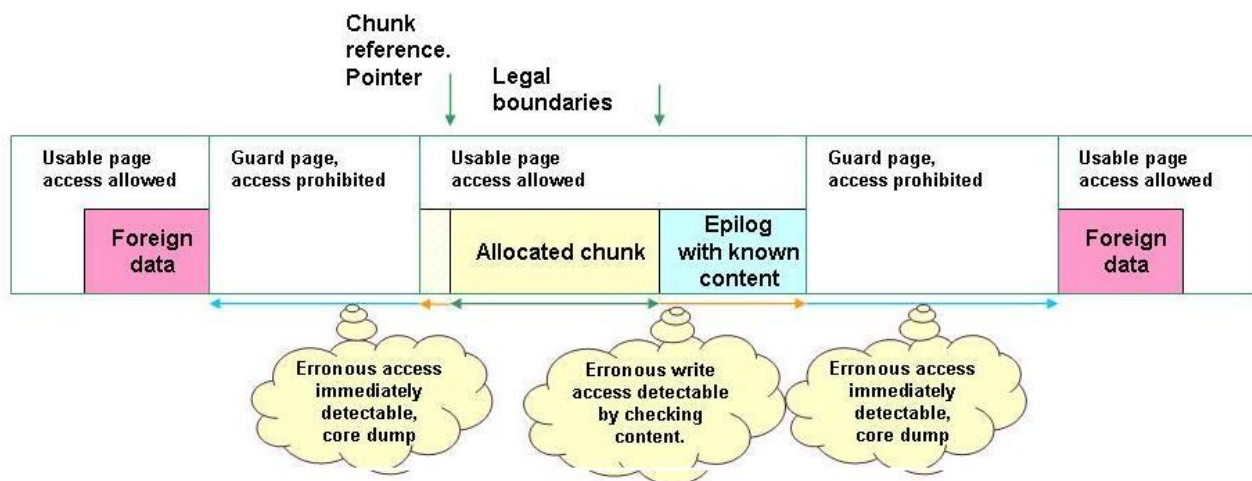
Heapchecker can frame each allocated chunk with

- eye catcher byte sequence or
- guard pages.

In the log example above the eye catcher damage was determined in a chunk of size 100 Byte at address 0x6608d90. Heapchecker can also dump the content of this memory chunk for analysis.

Guard pages technique is even more powerful than eye catchers. Every allocation will be placed between two guard pages protected by OS.

Eye catchers and guard pages techniques can be combined. The following picture points this up.



Conclusion

Heapchecker is a component of *SAP liballoc* library which allows to replace an origin (libc) memory management with another one developed at SAP.

Heapchecker is applicable in analysis of memory heap runtime issues. It can be used if analysis with higher level tools like e.g. ABAP Memory Analyzer was not possible or not successful.

Related Content

[Reference - Memory Management in ABAP AS](#)

[Reference - TREX MemWatch](#)

[Reference - dynamic linker and LD_PRELOAD](#)

For more information, visit the [Performance homepage](#).

Copyright

© Copyright 2010 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Excel, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, System i, System i5, System p, System p5, System x, System z, System z10, System z9, z10, z9, iSeries, pSeries, xSeries, zSeries, eServer, z/VM, z/OS, i5/OS, S/390, OS/390, OS/400, AS/400, S/390 Parallel Enterprise Server, PowerVM, Power Architecture, POWER6+, POWER6, POWER5+, POWER5, POWER, OpenPower, PowerPC, BatchPipes, BladeCenter, System Storage, GPFS, HACMP, RETAIN, DB2 Connect, RACF, Redbooks, OS/2, Parallel Sysplex, MVS/ESA, AIX, Intelligent Miner, WebSphere, Netfinity, Tivoli and Informix are trademarks or registered trademarks of IBM Corporation.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

SAP, R/3, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP Business ByDesign, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries.

Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius, and other Business Objects products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Business Objects S.A. in the United States and in other countries. Business Objects is an SAP company.

All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.