Using an alternative malloc implementation under Solaris to make better use of memory in memory intensive applications

William Schaub

Steuben Technologies

ABSTRACT

If you heavily use Solaris as a workstation and use various popular apps such as Firefox , OpenOffice , Acrobat reader and even gaim. Then you will notice that they heavily fragment and in many cases waste memory on Solaris eventually causing your system to start heavily swapping if you use them for long periods of time.

This document aims to help the reader compile and preload dlmalloc into their applications in order to avoid heavy paging and swapping when using these applications over a long term basis.

1. Introduction

I am a long time Solaris user. Not just on the server but on the desktop I run my workstation hard and almost never log out of CDE or quit my browser or office suite as I usually have many documents and web pages open in various workspaces. Each workspace is dedicated to a particular task I'm currently working on and they all might be in various stages of completion at any given time.

I rely on this to keep state on what I'm currently working on, the problem is under Solaris it seems that certain apps that I use heavily like OpenOffice and FireFox do not deal at all well with being open for long periods of time and seem to constantly grow until everything in the system starts to run out of resources and the application eventually even crashes.

Since the primary development platform for most of these tools is Linux I decided to try replacing the system malloc with Doug Lea's malloc since the glibc version of malloc is based on his malloc implementation and has been shown to perform well in conserving memory, and even has the ability to return memory to the operating system under certain conditions. This has worked out very well for me and I intend to share with you the fairly simple process of doing this yourself.

2. Getting dlmalloc up and running on your system

The first thing to do is read Doug Lea's web page on his memory allocator you can find it at http://g.oswego.edu/dl/html/malloc.html

This should tell you right away what this memory allocator is about and some very generic information on how to build it. There is also a lot of good information inside the source file itself.

2.1. fetching dlmalloc

Fetch the most recent dlmalloc from the URL below. The malloc.h file is only needed if you want to compile and link in Doug lea's malloc directly into your software and want to be able to use some of the extra built in functions (like mspace_malloc, mspace_...)

```
$ wget ftp://g.oswego.edu/pub/misc/malloc.c
$ wget ftp://g.oswego.edu/pub/misc/malloc.h
```

2.2. Compiling dlmalloc

By default dlmalloc will not compile as a thread safe library. Needless to say this will not be very useful at all for modern applications. Thankfully recent versions of the library have locks using POSIX threads which you can enable by defining *USE_LOCKS* and linking the *pthread* library into the shared object.

The following examples use the Sun Workshop compilers (or whatever sun calls them this week) aka SUNWSpro. but you can use gcc if you like. I recommend you take a quick look at docs.sun.com and pull up the ***Linker and Libraries Guide*** for your version of Solaris.

2.2.1. Compiling the 32 bit library

```
$ mkdir 32
$ cc -xO4 -DMSPACES -DUSE_LOCKS -o dlmalloc.so -G -K pic malloc.c -lpthread
$ mv dlmalloc.so 32
```

2.2.2. Compiling the 64 bit Library

```
$ mkdir 64
$ cc -xO4 -xarch=v9 -DMSPACES -DUSE_LOCKS -o dlmalloc.so -G -K pic malloc.c -lpthread
$ mv dlmalloc.so 64
```

3. Making use of dlmalloc in applications.

There are several ways you can make use of the newly compiled dlmalloc on your Solaris system.

- Link it into your applications when you compile them from source.
- Use LD_PRELOAD to pre-link it into an individual application.
- Set LD_PRELOAD into your .profile and force everything to link it in.

3.1. Installing dlmalloc.

The following example shows how I have set up dlmalloc on my system. I have chosen to install it inside of /usr/lib, /usr/lib/sparcv9 and /usr/lib/secure and /usr/lib/secure/64 so that I can run everything under my login with dlmalloc.so preloaded in. I simply set LD_PRELOAD in my .profile to LD_PRELOAD=dlmalloc.so and the runtime linker will find it in the normal system search paths for 32 and 64 bit libraries.

```
# install -f /usr/lib 32/dlmalloc.so
32/dlmalloc.so installed as /usr/lib/dlmalloc.so
# install -f /usr/lib/secure 32/dlmalloc.so
32/dlmalloc.so installed as /usr/lib/secure/dlmalloc.so
# install -f /usr/lib/sparcv9 64/dlmalloc.so
64/dlmalloc.so installed as /usr/lib/sparcv9/dlmalloc.so
# install -f /usr/lib/secure/64 64/dlmalloc.so
64/dlmalloc.so installed as /usr/lib/secure/64/dlmalloc.so
#
```

Once this is done you can set LD_PRELOAD=dlmalloc.so when ever you want to run an application with the new memory allocator. Or even set it in your profile and have your entire login session use the new allocator, which is quite handy If you have any problems you can still disable it and use the system malloc by unsetting LD_PRELOAD in the shell you are running the commands in.