fsmem: a library for memory augmentation using memory mapped files

Aaron D. Vose
Electrical Engineering and Computer Science
University of Tennessee
Knoxville, Tennessee 37919, USA
avose@utk.edu

Abstract

The "fsmem" (file system memory) library is designed to allow Linux applications that allocate memory based on the malloc() interface to access more memory than is available in system RAM. This is accomplished by memory mapping a large file on disk into the application's address space, and then allowing the `ptmalloc` allocator to carve up this large memory area into smaller blocks as needed.

1 Introduction

With the constantly increasing size of modern datasets, it is not uncommon for data intensive tools to run out of available memory on common desktops, workstations, and servers. For example, tools for building phylogenetic trees from multiple sequence alignments, such as PhyML [1] and RAxML [2], can require tens to hundreds of gigabytes of memory to run on data sets involving only a few thousand species. While ten gigabytes of RAM is not uncommon for high-end workstations, hundreds or thousands of gigabytes of RAM is simply not available in anything but the most expensive high-end machines currently available. Thus, processing large datasets with common tools is often impossible for many users.

A library has been created to allow these data and memory intensive applications to access orders of magnitude more memory than was possible in the past, through the use of large file-based back-ends. While Linux swap partitions can be used to increase the amount of memory available, doing so has some important drawbacks that do not apply to the file-based approach. First, swap partitions typically need to be created and setup by a system administrator with root access to the machine, while files can be created and managed by regular accounts with only user-level permissions. Second, allocating space to swap partitions is a heavy operation that requires committing disk resources to swap on a longer-term basis compared to the speed and ease with which files can be created and removed.

2 Implementation

The current version of the fsmem library implements a very simple and fast form of a file-based back-end. First, fsmem creates a single large file (defaults to 1 TB in size) at application start, memory maps it into the application's address space and then hands this single large block to version 3 of the `ptmalloc` allocator which carves off chunks from this block as needed to satisfy malloc() requests. The `ptmalloc` allocator is a very fast and memory-efficient allocator which has been used for years as a part of the core GNU/Linux libraries [3].

Accessible means exist to mitigate the performance issues associated with running off disk instead of RAM. First, the operating system will use RAM as a cache for the disk-backed memory, by virtue of its efforts to keep commonly accessed memory pages in RAM. Additionally, the use of a RAID0 array of fast disks such as SSDs (solid state disks) can drastically improve throughput and random access speeds. For comparable price, SSDs provide an order of magnitude more memory capacity than DRAM at a performance level that is acceptable for many applications.

3 Using fsmem

The fsmem library can be obtained online from `http://www.aaronvose.com/fsmem`. Once downloaded, there are three steps to using the library with a desired malloc()-based application: (1) Compiling / building the file-system back-end code, (2) Compiling / building the `ptmalloc` block allocator, and (3) Linking the produced `libptmalloc3.a` file with the application needing access to large amounts of memory. It is important to note here that the application needing access to large amounts of memory does not need to be rewritten to use a different memory allocation API to take advantage of the increase in available memory. Simply linking with the produced `libptmalloc3.a` file will cause the application to use the fsmem back-end to satisfy malloc() requests.

- Download the fsmem package:
  ```
  > wget http://www.aaronvose.com/fsmem/fsmem-9-16-2012.tar.gz
  ```
- Untar the package:
  ```
  > tar -zxvf fsmem-9-16-2012.tar.gz
  ```
- Build the fsmem back-end:
  ```
  > cd fsmem-9-16-2012
  > make
  ```
- Build the `ptmalloc` block allocator:
  ```
  > cd ptmalloc3
  > make
  ```
- Link your application against the produced `libptmalloc3.a` file.

Acknowledgments

This work was supported in part by the DOE Office of Science, Office of Biological and Environmental Research, through the BioEnergy Science Center (BESC), a DOE Bioenergy Research Center.

References

[1] S. Guindon, J. Dufayard, V. Lefort, M. Anisimova, W. Hordijk, and O. Gascuel, "New algorithms and methods to estimate maximum-likelihood phylogenies: assessing the performance of phyml 3.0," *Systematic biology*, vol. 59, no. 3, pp. 307–321, 2010.

[2] A. Stamatakis, T. Ludwig, and H. Meier, "Raxml-iii: a fast program for maximum likelihood-based inference of large phylogenetic trees," *Bioinformatics*, vol. 21, no. 4, pp. 456–463, 2005.

[3] "Wolfram gloger's malloc homepage." http://www.malloc.de/en/. Accessed: Aug/9/2012.