TABLE OF CONTENTS

Introduction........................................................................................................................................... 3
Why Hoard?........................................................................................................................................ 3
Contention......................................................................................................................................... 3
False Sharing................................................................................................................................. 3
Blowup............................................................................................................................................... 3
How Do I Use Hoard?......................................................................................................................... 4
Who’s Using Hoard?.......................................................................................................................... 4
Building Hoard................................................................................................................................ 5
Generic Builds.................................................................................................................................. 5
Windows Builds............................................................................................................................... 5
Using Hoard....................................................................................................................................... 6
UNIX................................................................................................................................................ 6
Solaris................................................................................................................................................ 6
Linux................................................................................................................................................ 6
Windows.......................................................................................................................................... 7
Using libhoard .................................................................................................................................. 7
Using Detours.................................................................................................................................. 7
License Information........................................................................................................................... 8
More Information............................................................................................................................... 8
Feedback & Discussion...................................................................................................................... 9
Mailing lists....................................................................................................................................... 9
**Introduction**

The Hoard memory allocator is a fast, scalable, and memory-efficient memory allocator for shared-memory multiprocessors. It runs on a variety of platforms, including Linux, Solaris, and Windows.

**Why Hoard?**

**Contention**

Multithreaded programs often do not scale because the heap is a bottleneck. When multiple threads simultaneously allocate or deallocate memory from the allocator, they will be serialized. Programs making intensive use of the allocator actually slow down as the number of processors increases. Your program may be allocation-intensive without you realizing it, for instance, if your program makes many calls to the C++ Standard Template Library (STL).

**False Sharing**

The allocator can cause other problems for multithreaded code. First, it can lead to *false sharing* in your application: threads on different CPUs can end up with memory in the same cache line, or chunk of memory. Accessing shared cache lines can be hundreds of times slower than accessing unshared cache lines.

**Blowup**

Multithreaded programs can also lead the allocator to *blowup* memory consumption. This effect can multiply the amount of memory needed to run your application by the number of CPUs on your machine: four CPUs could mean that you need four times as much memory.

Hoard is a fast allocator that solves all of these problems.
**How Do I Use Hoard?**

Hoard is a drop-in replacement for `malloc()`, etc. In general, you just link it in or set just one environment variable. You do not have to change your source code in any way. See the section “Windows Builds” below for more information for particular platforms.

**Who’s Using Hoard?**

Users of Hoard include **AOL**, **British Telecom**, **Crystal Decisions**, **Entrust**, **Novell**, **Coyote Systems** (for their BEMEngine product) and **OpenWave Systems** (for their Typhoon & Twister servers). Open source projects using Hoard include **Ardour**, the **Bayonne** GNU telephony server and the GNU **Common C++** system.
Building Hoard

**Generic Builds**

You can use the available pre-built binaries or build Hoard yourself. Hoard is written to work on Windows and any variant of UNIX that supports threads, and should compile out of the box using `make`. You may need to choose your operating system as follows:

```make
make USE_LINUX=1
make USE_WINDOWS=1
make USE_SOLARIS=1
make USE_HPUX=1
```

See `Makefile.common` for more options.

Note: you *must* use the GNU version of `make` to build Hoard. This version of make ships standard with Linux and is usually installed on Solaris systems. There is a Windows version in the Hoard distribution called `gnumake.exe` in the `misc` directory. The source for GNU make is always available at `ftp://ftp.gnu.org/pub/gnu/make/`.

**Windows Builds**

There are now two alternative ways of using Hoard with Windows. The default approach above builds `libhoard.dll` and its associated library `libhoard.lib`. The second approach relies on Microsoft Research’s Detours ([http://research.microsoft.com/sn/detours](http://research.microsoft.com/sn/detours)). With Detours, you can take advantage of Hoard without having to relink your applications. Install Detours into `C:\detours`, and then build the Hoard detours library:

```make
make USE_WINDOWS=1 hoarddetours.dll
```
Using Hoard

UNIX

In UNIX, you can use the `LD_PRELOAD` variable to use Hoard instead of the system allocator for any program not linked with the `-static` option (that's most programs). Below are settings for Linux and Solaris.

Solaris

```bash
setenv LD_PRELOAD "/path/to/libhoard.so/usr/lib/libthread.so/usr/lib/librt.so/usr/lib/libCrun.so.1"
```

*Note:* For some security-sensitive applications, Solaris requires that libraries used in `LD_PRELOAD` be placed in the `/usr/lib/secure` directory. In that event, after copying these libraries into `/usr/lib/secure`, set `LD_PRELOAD` as below:

```bash
setenv LD_PRELOAD "libhoard.so:libthread.so:librt.so:libCrun.so.1"
```

Linux

```bash
setenv LD_PRELOAD "/usr/lib/libdl.so/path/to/libhoard.so/usr/lib/libpthread.so"
```

A Debian package for Hoard is available at packages.debian.org/libhoard.
Windows

You can now use Hoard in two ways on Windows.

Using libhoard

When you build Hoard under Windows, you'll get two files: libhoard.dll and libhoard.lib. Put the following into your source code as the very first lines:

```c
#if defined(USE_HOARD) && defined(_WIN32)
#pragma comment(lib, "libhoard.lib")
#endif
```

The best approach is to put this stanza into the first part of a header file included by all of your code. The pragma ensures that Hoard loads before any other library (you will need libhoard.lib in your path).

When you execute your program, as long as libhoard.dll is in your path, your program will run with Hoard instead of the system allocator.

Note that you must compile your program with the /MD flag, as in:

```
cl /MD /G6 /Ox /DUSE_HOARD=1 myprogram.cpp
```

Hoard will not work if you use another switch (like /MT) to compile your program.

Using Detours

By using Detours, you can take advantage of Hoard’s benefits without relinking your Windows application (as long as it is dynamically linked to the C runtime libraries). With this approach, Hoard performs memory allocation operations for small objects (< 8K) in your application. Larger objects will continue to be managed via the Windows allocator: the original allocation instructions will be executed (i.e., they are not bypassed).

You will need to use one of two of the Detours tools (setdll.exe or withdll.exe) in conjunction with this version of Hoard. To temporarily use Hoard as your allocator, use withdll:

```
withdll -d:hoarddetours.dll myprogram.exe
```
If you want your program to always use Hoard, you can use setdll to add it to your executable:

```bash
setdll -d:hoarddetours.dll myprogram.exe
myprogram.exe
```

You can always remove Hoard from your executable by using the `-r` option of setdll.

**License Information**

The use and distribution of Hoard is governed by a modified version of the GNU Lesser General Public License version 2.1 as published by the Free Software Foundation, http://www.fsf.org: see the included file COPYING for more details.

Notice that distribution of Hoard requires that you provide a link to the Hoard website in your program's documentation and/or that you include the source code for Hoard. Any modifications you make to the Hoard source code must be provided to me. If you statically link your code with Hoard, you must provide complete object files for your code to the recipients. This license further requires that, if you include the source code of Hoard in your source base, you must make your entire source base publicly available. If you distribute your program with Hoard, you must "show" your users these licensing terms as described in the file COPYING.

Alternatively, you may purchase a commercial license for Hoard that does not include the above restrictions. Please visit http://www.hoard.org for more information.

**More Information**

For technical information on Hoard along with some nice performance graphs, read the following paper included in docs/asplos2000.pdf.


The latest version of Hoard is always available at http://www.hoard.org.
Feedback & Discussion

Please send any bug reports and other feedback to emery@cs.umass.edu.

Mailing lists

There are two mailing lists for Hoard:

hoard-announce@yahoogroups.com:
a low-volume mailing list for announcements
of new releases of Hoard. Only I can post to this list.

hoard@yahoogroups.com:
a mailing list for Hoard-related discussions.
Any member of the mailing list may post to it.

To subscribe, just send a message to the mailing list address to get further instructions on subscribing.

Emery Berger
Assistant Professor
Department of Computer Science
University of Massachusetts
Amherst, MA 01003

www.cs.mass.edu/~emery
emery@cs.umass.edu