HoloStor 1.0 Library for Linux

Release Notes

Version 1.0.4

Thomas P. Scott <tpscott@alum.mit.edu>
Myron Zimmerman <MyronZimmerman@alum.mit.edu>

HoloStor 1.0 Library for Linux

Release Notes

Version 1.0.4

Table of Contents

This page is intentionally blank.

About the HoloStor 1.0 Library

The HoloStor 1.0 Library enables OEMs of storage devices, whether clustered RAIN (Redundant Array of Independent Nodes) or traditional RAID devices, to achieve higher levels of reliability at lower cost than any other method possible today. HoloStor can also be implemented in an ASIC to support hardware-based RAID controllers. The core technology takes the data that needs to be protected, breaks it up into groups of N blocks and calculates K redundancy blocks for each group. When these N + K blocks are dispersed to N + K storage media by the OEM, they provide data protection from up to K storage media failures. The key property of this encoding technology is such that the entire data set can be recreated with **any** N of the N + K blocks. Thus the storage device can withstand the loss of K nodes without any loss of data.

The various steps performed by this library that are relevant to storage devices are:

1. Encoding of the redundancy blocks by the HoloStor_Encode() function.

2. Decoding of the data and redundancy blocks by the HoloStor_Decode() function, if needed, to recover any missing blocks. Please note that if there is no loss of any nodes, then the decoding step is not needed.

The library makes use of a patented data coding method that has been optimized for operation on Pentium compatible processors. The library determines which instructions are supported and will take advantage of SSE2 and MMX instructions to provide high performance.

The HoloStor 1.0 Library for Linux contains the following:

- The interface specification for the library.

- The HoloStor.h header file defining the programming interfaces for C/C++.

- The HoloStorLib.a static library which implements the Encoding and Decoding algorithms in user space.

- The HoloStorMod.o link module which implements the Encoding and Decoding algorithms in kernel space.

- A sample program for illustrating the use of HoloStor in user space and kernel space and for measuring library performance.

System Requirements

To use the HoloStor 1.0 Library for development, you must have:

- A PC-compatible system with the Linux operating system installed. The HoloStor 1.0 Library should be compatible with any recent release of Linux. But most testing for this release has been with Ubuntu 10.10.

- GNU gunzip utility to uncompress the installation package.

- Adobe Acrobat Reader to view the Interface Specification and these release notes.

- GNU gcc and make packages to build the sample program as well as your application. Version 4.4.5 of gcc was used to build the HoloStor 1.0 Library and is recommended for applications that use it.

Installation

The library, code sample, Interface Specification and these Release Notes are provided as a GNU zipped tar file (i.e. a tarball).

Installing HoloStor 1.0 Library

To install HoloStor 1.0 Library:

1. Expand the files in a directory that you wish to do your development:
   `tar xvzf HoloStorLib-linux-<arch>-1.0.4.tar.gz`
   where <*arch*> is i686 and x86_64 for 32- and 64-bit Linux, respectively.

2. Installation is complete.

Verifying the Library Version (Optional)

The HoloStor.h include file defines the HOLOSTOR_VERSION macro that identifies the library version for programs compiled with the library.

The prebuilt library has an embedded copyright string that can be used to identify the version of the library and the programs linked with it.

To display the version string, run the *strings* command:
`strings HoloStorLib.a` | grep –i copyright

The command will return the following string for this version of the library:

```
HoloStor 1.0.4 Copyright (C) 2003-2011 Thomas P. Scott and Myron Zimmerman
```

Uninstalling HoloStor 1.0 Library

To uninstall HoloStor 1.0 Library, simply delete the installed files.

Table of Installed Files and Directories

The HoloStor 1.0 Library consists of the files listed in Table 1.

Table 1. Installed Files.

File	Description
HoloStorLib.a	The HoloStor 1.0 Library as a static library.
HoloStorMod.o	The HoloStor 1.0 Library as a kernel linkable module.
HoloStor.h	The HoloStor 1.0 include file.
HoloStorLibReleaseNotes.pdf	These release notes.
HoloStorInterfaceSpec–1.0.pdf	The interface specification for the HoloStor 1.0 Library interface.
Samples/Makefile	Makefile for building the EncodeDecode code sample for use in Linux user or kernel space.
Samples/Makefile-LKM	Supplemental Makefile for building the EncodeDecode code sample as a Linux loadable kernel module (LKM).
Samples/EncodeDecode.c	Source for the EncodeDecode code sample.
Samples/wrapper.c	Source for support routines for running the EncodeDecode code sample in kernel space.
Samples/EncodeDecode.exe	Pre-built EncodeDecode sample for execution in user space.
Samples/EncodeDecode.ko	Pre-built EncodeDecode sample for execution as a Linux 2.6.35 LKM.[1]

Code Sample

The HoloStor 1.0 Library includes a code sample, EncodeDecode.c, which illustrates the use of the library interfaces and can be used to benchmark library performance. The following sections describe the code sample and the steps necessary for building and running the code sample.

About the EncodeDecode Sample

The EncodeDecode sample program is designed to repetitively encode test data, simulate all possible combinations of failing nodes and decode the data. The program will repetitively test over a range of block sizes, a range of data blocks numbers and a range of ECC block numbers.

[1] The pre-built version of EncodeDecode for kernel spaces is specific to a particular release of Linux. Use the Makefile to create a version that corresponds to your release of Linux.

Coding performance is calculated from Pentium cycle measurements performed by the RDTSC instruction. To help simulate real world performance, the program provides user options for preparing the cache before each performance measurement. Consequentially, both cold and warm cache performance numbers can be measured.

The EncodeDecode sample supports a number of runtime options. These options and their defaults are described in Table 2. When running EncodeDecode in user space, options are case insensitive. When running EncodeDecode in kernel space, options are case sensitive.

Table 2. EncodeDecode command line options. Parameter *n* refers to a user supplied integer argument and *hex* refers to a user supplied hexadecimal argument.

Option	Default	Meaning
`/?`	n/a	Display usage.
`MinBsize=`*n*	1024	Specify the low bound, *n*, of the range of block sizes to test. Block sizes are in bytes. Minimum value of 64.
`MaxBsize=`*n*	1024	Specify the upper bound, *n*, of the range of block sizes to test. Block sizes are in bytes. Maximum value of 65536.
`MinData=`*n*	14	Specify the lower bound, *n*, of the range of data blocks per reliability group to test.
`MaxData=`*n*	14	Specify the upper bound, *n*, of the range of data blocks per reliability group to test. Maximum value of 17.
`MinEcc=`*n*	3	Specify the lower bound, *n*, of the range of ECC blocks per reliability group to test.
`MaxEcc=`*n*	3	Specify the upper bound, *n*, of the range of ECC blocks per reliability group to test. Maximum value of 4.
`Verbosity=`*n*	1	Specify the volume of output as an integer in the range 0-2. Higher numbers increase the volume of output.
`TestData=`*hex*	0	Specify the hexadecimal 32-bit word to fill buffers with. 0 means to use random fill.
`Cache=`*n*	0	Specify the type of cache preparation prior to benchmarking with integer *n*: 0 is no preparation, 1 is dirty fill the cache, 2 is clean flush the cache.

When iterating over the range of Data and ECC blocks per reliability group, the iteration is arithmetic with increments of 1. When iterating over the range of block sizes, the iteration is geometric with a multiplication factor of 2.

This version of the HoloStor library limits the total number of data plus ECC blocks to 17, and the number of ECC blocks to 4. Test cases for configurations which exceed these limits will simply report “HoloStor_CreateSession” failures.

Running the Sample in User Space

A pre-built executable of EncodeDecode for user space is provided in the Samples directory.

The usage for running the EncodeDecode sample in user space is:

```
EncodeDecode.exe [options]
```

The options and their defaults are described in Table 2.

Building the Sample

To rebuild the EncodeDecode sample for user space, change to the Samples directory and type `make release`. The result of this build is a statically linked executable with the name EncodeDecode.exe.

To build the EncodeDecode sample for kernel space, change to the Samples directory and type `make kernel`. The result of this build is a loadable kernel module with the name EncodeDecode.ko.

Running the Sample in Kernel Space

The usage for running the EncodeDecode sample in kernel space is:

```
insmod EncodeDecode.ko [cmdline=options]
```

This will load the module into kernel space. Root permission is required. Once loaded, the EncodeDecode program is run as the module's *init_module* function.

After EncodeDecode has successfully run, it must be unloaded:

```
rmmod EncodeDecode
```

The output of EncodeDecode is through the kernel's *printk* function and is consequently directed to both the console and to `/var/log/messages`. The command `dmesg` can be used to retrieve and display this output.

Options are passed to EncodeDecode through the module parameter `cmdline`. If more than one option is specified, *options* must be a quoted string to protect the spaces separating the options.

The options and their defaults are described in Table 2.

Programming Considerations

HoloStor is provided as at static library for use by programs compiled for user space and as a linkable module for use with Linux loadable kernel modules.

The user library implementation consists of the HoloStor.h header and the static library HoloStorLib.a. The library can be used with both C and C++ programs running in user space. Source files using the library interface need to include HoloStor.h. Objects using the library need to be linked with HoloStorLib.a. By default, the library allocates memory using the routines malloc() and free(). These defaults may be overridden by exporting definitions for the following prototypes from an object that is linked prior to the library:

```
void* HoloStor_QuickAlloc(unsigned int size);
void  HoloStor_QuickFree(void *p);
void* HoloStor_TableAlloc(unsigned int size);
void  HoloStor_TableFree(void *p);
```

The kernel implementation consists of the HoloStor.h header and the linkable module HoloStorMod.o. The Linux 2.6 kernel uses different subroutine calling conventions than those used in user space and consequently the need for a kernel specific implementation of HoloStor. There are no default definitions for the memory allocation routines used by HoloStorMod.o and these definitions must be provided in an object that is linked with the module. Example definitions follow:

```
#include <linux/slab.h>
#include <linux/vmalloc.h>
//
void* HoloStor_QuickAlloc(unsigned int size) {
       return kmalloc(size, GFP_KERNEL);
}
void  HoloStor_QuickFree(void *p) { kfree(p); }
void* HoloStor_TableAlloc(unsigned int size) {
       return vmalloc(size);
}
void  HoloStor_TableFree(void *p) { vfree(p); }
```

See the Makefile and wrapper.c code samples for additional details regarding the operation of HoloStorMod.o in kernel space.

Data and ECC buffers on systems with processors with SSE2 instructions (e.g. Pentium 4) must be aligned on 16-byte boundaries. In general, buffers should be aligned on cache line boundaries (e.g. 64 bytes on Pentium 4 processors) for maximum performance. Examples of how to align static data buffers are provided in the code samples.

New Features and Updates

This release of HoloStor 1.0 is open source. In addition to being open source, this release adds support for x86_64 architecture (i.e. 64-bit support) and refreshes support for Linux 2.6 Loadable Kernel Modules.

Some changes to the HoloStor interface specification were made as part of adding 64-bit support. This version of the HoloStor 1.0 Library complies with HoloStor Interface Specification 1.0a, dated May 15, 2011.

Notes and Known Limitation

The following are open issues in this release of the HoloStor 1.0 Library for Linux:

- Operating System preemption of “EncodeDecode” execution due to hardware interrupts or context switch to other processes, will add significant CPU cycles to timing values displayed by “EncodeDecode”. A very high maximum value typically indicates such an occurrence.

- The EncodeDecode.ko module does not have a GPL license string and `insmod` will consequently display a warning that the loading of EncodeDecode.ko will taint the kernel. This warning can be disregarded.