Building and Linking Static Versions of Memkind and Numa

Published 11/16/2016

Abstract

This paper offers a solution for building and linking static versions of libraries. A script demonstrating building static versions of libnuma and libmemkind is provided as an example.

Contents

Introduction

The memkind library provides ways to access and manage the MCDRAM memory on KNL processors. The memkind library and its dependency, libnuma, are provided as dynamic libraries by Intel in its XPPS packages[1]. Cray, in turn, passes these through in the CLE 6.0 releases[2] and provides a module file, cray-memkind, that makes it easy for users to link in these libraries. Some users may prefer to link statically which requires static versions of these libraries. This white paper provides instructions on how static versions can be built and how a new modulefile can be created.

[1] XPPS is the new name for the Xeon Phi Processor Software package provided by Intel. It was formerly known as MPSP.
[2] Memkind is expected to be added to the SLES distribution at some time in the future. When that happens, CLE will pass through the version of memkind that comes with the SLES distribution.

Creating and Linking Static Libraries

The following script can be used to create a directory containing static versions of the libnuma.a and libmemkind.a libraries for a user on the system. The script demonstrates cloning the respective libraries from Github, building static versions of libnuma.a and libmemkind.a, and installing them in "components/lib". As an alternative to Github, libraries can be pulled from Intel's repository.

> Note: when using static versions of numa and memkind, these libraries must be linked with whole-level archive into your application. Add the following arguments when building an executable:
> `-Wl,--whole-archive –lnuma –lmemkind –Wl,--no-whole-archive`

Script Prerequisites:

- Autoconf 2.69 or higher
- Automake 1.15 or higher

```
#!/bin/bash

# Requires Autoconf 2.69 (or higher)
# Requires Automake 1.15 (or higher)
# Applications must be linked with
# -Wl,--whole-archive -lnuma -lmemkind -Wl,--no-whole-archive
# so that the optional symbols are included in the binary

# Set the default paths
# install variable is where the libraries will be installed, default "components/lib"
install=components

# build variable is where the libraries are built, default "build/"
build=build

# use the name memkind for the memkind git clone
memkind_git=memkind

# use the name numactl for the numactl git clone
numa git=numactl

# make the directories
mkdir -p ${install}
mkdir -p ${build}

# get the full paths
installpath=$(readlink -e ${install})
buildpath=$(readlink -e ${build})

# clone the repos if they don't already exist
cd ${buildpath}
# Memkind has a rapidly evolving API, The version used on Cray is 1.0.0
# Using versions newer than 1.0.0 may prevent the usage of Memkind features
[ -d ${memkind_git} ] || git clone http://github.com/memkind/memkind.git ${memkind_git} -b v1.0.0
[ -d ${numa_git} ] || git clone https://github.com/numactl/numactl.git ${numa_git}

# build libnuma
cd ${buildpath}/${numa_git} && \
        ./autogen.sh && \
        ./configure --prefix=${installpath} && \
        make && make install

# build memkind
# WARNING - the build_jemalloc.sh script is fragile, and only runs on a clean clone
cd ${buildpath}/${memkind_git} && \
    ./build_jemalloc.sh && \
    ./autogen.sh && \
    ./configure --prefix=${installpath} --with-numa=${installpath} && \
    make && make install
```

Modulefile Support

Some users may wish to have a modulefile and a pkg-config file to facilitate setting up their environment to use the static versions of memkind and libnuma that they have created. An example of how this can be done is shown below.

In an accessible, writeable directory, add a modulefile similar to the one shown below. For purposes of these examples, the name of the file is assumed to be memkind-static. The name of the location of the static libraries, stored in the location_static variable, should be modified to indicate the location where the static memkind and numa libraries are stored.

```
#%Module
#
# Example modulefile for static memkind library
#
#

# Set destination location for the static libraries
set location_static = /lus/dal/memkind/lib

proc ModulesHelp { } {
        puts stderr "Provides static versions of libmemkind and libnuma"
}

# use conflict statement to prevent user from loading the dynamic memkind module
conflict cray-memkind

# Put the library path in
prepend-path    LD_LIBRARY_PATH ${location_static}
prepend-path    LIBRARY_PATH        ${location_static}

# the static versions of libnuma and libmemkind must be linked with
# whole-archive to bring in the optional symbols
#
# This is pulled in via pkgconfig

append-path     PKG_CONFIG_PATH         {location_static}/pkgconfig
prepend-path    PE_PKGCONFIG_LIBS       memkind-static
```

Pkg-config

A pkg-config file will allow the linker to automatically include the static versions of the libraries and place the whole-archive and no-whole-archive settings in the proper locations in the link command. The name of the pkg-config file should match the name in the `PE_PKGCONFIG_LIBS` variable with the `.pc` extension as set in the modulefile (e.g., `memkind-static.pc`). Store this file in the directory indicated in the `PKG_CONFIG_PATH` variable set in the modulefile (e.g., `/lus/dal/memkind/lib/pkgconfig`).

Example pkg-config file

```
Name: memkind-static

Description: Intel memkind

Version: 1.0

Cflags:

Libs: -Wl,--whole-archive -lnuma -lmemkind -Wl,--no-whole-archive
```