DASHMM Installation Guide

The Dynamic Adaptive System for Hierarchical Multipole Moments (DASHMM) is a C++ library providing a general framework for computations using multipole methods. In addition to the flexibility to handle user-specified methods and expansion, DASHMM includes built-in methods and expansions, including the Barnes-Hut (BH) and Fast Multipole Method (FMM), and two expansions implementing the Laplace Kernel used in electrostatics and Newtonian gravitation. Although only the Laplace kernel is implemented in DASHMM, in future versions more kernels will be added.

DASHMM is built using the advanced runtime system, HPX-5, but the basic interface to DASHMM does not require any knowledge of how to use HPX-5. Instead, the basic interface handles the distribution of the parallel work. The model of use for the basic interface to DASHMM is to take a serial code, make some calls to the DASHMM library, and in so doing, the program will make use of parallel resources. In this mode of operation, when execution is outside a DASHMM library call, the runtime can be considered to be not operating. Data saved in the global address space provided by HPX-5 persists between calls, but the runtime is not executing any operations. More options are available in the DASHMM Advanced User Guide, including interoperability with legacy MPI codes, or use in HPX-5 aware applications.

In the following, snippets of code, or the names of code constructs will be set in a fixed width font. For example, `main()`.

This document covers the installation of the DASHMM library. For instructions on using the library, please see the DASHMM Basic User Guide and the DASHMM Advanced User Guide. The latest information, resources and tutorials can be found at the DASHMM webpage: https://www.crest.iu.edu/projects/dashmm/

Prerequisites

DASHMM depends on one external library: HPX-5. The current version of DASHMM depends on version 2.1.0 of HPX-5, which can be found at https://hpx.crest.iu.edu/

The current version of DASHMM is specialized to SMP operation, so there is little to be specified in the configuration of HPX-5 for use with this version of DASHMM. Assuming that you have unpacked the HPX-5 source into the folder `/path/to/hpx/` and wish to install the library into `/path/to/install/` the following steps *should* build HPX-5 and install it.

1. ‘`cd /path/to/hpx/`’
2. ‘`./configure –prefix=/path/to/install`’
3. ‘`make`’
4. ‘`make install`’

The previous commands will build HPX-5 and install it into the given location. This configuration uses many of the default HPX-5 options, and may not be ideal for specific system. Please see the official HPX-5 documentation for more details.

The DASHMM build system relies on the `pkg-config` utility to specify the needed HPX-5 compilation and linking options, so it is important to add the correct path for HPX-5 to your `PKG_CONFIG_PATH`. While one is at it, it is useful to modify the following environment variables to point to the newly installed HPX-5. For example, with `bash`:

```
export PATH=/path/to/install/bin:$PATH
export LD_LIBRARY_PATH=/path/to/install/lib:$LD_LIBRARY_PATH
export PKG_CONFIG_PATH=/path/to/install/lib/pkgconfig:$PKG_CONFIG_PATH
```

Installation

The DASHMM library is straightforward to build. Once the previous prerequisites are met, one needs to perform the following steps:

1. Unpack the source code into some convenient directory. For the sake of discussion, this guide assumes that the code has been unpacked in `/path/to/dashmm/`.

   Change to the DASHMM directory into which you have unpacked the code. There are a number of subfolders. Initially, the most relevant of these are `doc/` and `demo/`.
2. In `/path/to/dashmm/` can be found `Makefile`. There are few (if any) changes that need to be made to this file for successful compilation. The most likely change to make would be to modify the compiler used. Any compiler supported by HPX-5 will be able to compile DASHMM, provided it also supports the C++11 standard.

   For example, to change the compiler from the default (g++) to the Intel C++ compiler, one needs to replace `CXX = g++` with `CXX = icpc`.
3. Run `make` from `/path/to/dashmm/`. This should build the library statically, and it will be ready to link with your code.

Test Code

Included with DASHMM is a test code that demonstrates a simple use of the library. This code is given in `/path/to/dashmm/demo/basic/`. It can be built by running `make` in the `demo/basic/` folder once the library has been built. The simple `Makefile` in `demo/basic/` is an example of how to link your code with the DASHMM library. For more description, see the next section.

The test code creates a random distribution of source and target points and computes the potential at the targets due to the sources using the Laplace kernel, using any of the currently available built-in methods in DASHMM. A user can request a summary of the options to the program by running the code with `--help` as a command line argument, or by reading `/path/to/dashmm/demo/basic/README`.

Linking against DASHMM

To build a program using the DASHMM library, only a few things need to be done. DASHMM builds in place, so when compiling code that uses the library, one must specify where to look for the header files, and where to look for the built library. Further, because DASHMM relies on HPX-5, one must also specify how to find HPX-5. For HPX-5 this is easiest with the `pkg-config` utility.

Assuming that DASHMM was built in /path/to/dashmm/ then to compile code (with, for example, `g++`) one must specify the following arguments for compilation:

```
-I/path/to/dashmm $(shell pkg-config --cflags hpx)
```

Similarly, one must specify the following arguments for linking:

```
-L/path/to/dashmm -ldashmm $(shell pkg-config --libs hpx)
```

Examples of this can be found in the demo programs included with DASHMM.