The distcc Utility for Remote Compiling

Fall 2017

Department of Mechanical Engineering, CSAIL
MIT, Cambridge MA 02139

1 The distcc Utility for Remote Compiling

distcc is a program to distribute builds of C, C++, Objective C or Objective C++ code across several machines on a network. distcc should always generate the same results as a local build, is simple to install and use, and is normally much faster than a local compile. The following instructions were borrowed and then modified for Mac OS X from Josh Leighton at:

https://wikis.mit.edu/confluence/display/hovergroup/Duovero+Setup

1.1 Raspberry Pi Setup

In the following subsections we will ensure that the correct version of GCC and G++ are installed on the Raspbian and install distcc. In order to start using distcc it must be setup as the default compiler. When compiling a program, distcc will attempt to look for a more powerful host. If a host is not found, then distcc will compile locally with native compiler.

1.1.1 Updating Raspbian and Installing Correct Compiler Versions

Let's first make sure Raspbian packages are up to date.

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

We will install gcc/g++ version 4.8. It is vital that the compiler versions on the Raspbian for both gcc and g++ match the versions that will be used on the Mac OS X machines. At the time of this writing, pre-compiled versions of gcc/g++ 4.8 and 4.9 are available for cross-compiling on the Mac OS X, described below. Raspbian Wheezy, based on kernel version 3.18, by default has versions 4.6. Raspbian Jessie, based on kernel version 4.1, by default has versions 4.9. As a good middle ground we will install gcc/g++ version 4.8 as both Wheezy and Jessie suppport it.

```
$ sudo apt-get install gcc-4.8 g++-4.8
```

At this point the symlinks in `/usr/bin` point to version 4.6 or 4.9, depending on the version of Raspbian you have. We must remove those symlinks and create new ones pointing to version 4.8.

```
$ cd /usr/bin
$ sudo rm g++
$ sudo rm gcc
$ sudo ln -s /usr/bin/g++-4.8 /usr/bin/g++
$ sudo ln -s /usr/bin/gcc-4.8 /usr/bin/gcc
```

1.1.2 Install distcc

At this point we have ensured that `gcc` and `g++` versions match what is available for the Mac OS X. Now, let's install `distcc` with all the proper system variables.

```
 $ sudo apt-get install distcc
```

Double check that the following symlinks are in `/usr/lib/distcc` and if not create them.

```
$ cd /usr/lib/distcc
$ ls -la arm-linux-gnueabihg-g++-4.8
arm-linux-gnueabihf-g++-4.8 -> ../../bin/distcc
```

and

```
$ ls -la arm-linux-gnueabihg-gcc-4.8
arm-linux-gnueabihf-gcc-4.8 -> ../../bin/distcc
```

Let's ensure that when compiling that distcc is called by modifying `.bashrc` to include these lines:

```
export CC=/usr/lib/distcc/arm-linux-gnueabihf-gcc-4.8
export CXX=/usr/lib/distcc/arm-linux-gnueabihf-g++-4.8
export DISTCC_HOSTS=192.168.2.1 # replace this address with that of the desired host
```

The `DISTCC_HOSTS` can have a white space separated list of different hoosts to try. For example,

```
export DISTCC_HOSTS=192.168.1.255 192.168.1.144
```

The following `DISTCC` configuration parameters ensure compilation happens off of the Raspberry Pi and on the more powerful slave machines.

```
export DISTCC_BACKOFF_PERIOD=0
export DISTCC_IO_TIMEOUT=3000
export DISTCC_SKIP_LOCAL_RETRY=1
```

1.2 Host Mac OS X Setup

Compared to the Raspberry Pis, the Mac OS X machines available are much more powerful. The only thing needed to setup `distcc` is obtaining the ARM toolchain with the corresponding versions of `gcc` and `g++` used on the Raspberry Pis and setting up `distcc` as a local daemon to accept incoming requests.

The fastest way to obtain the ARM toolcahin for Mac OS X is finding it already compiled such as from this site: http://www.welzels.de/blog/en/arm-cross-compiling-with-mac-os-x/

From within the site, choose and download the `.dmg` with the appropriate `gcc/g++` versions. Open and run the `.dmg` installer. Specifically, we are looking for the Raspberry Pi and versions 4.8.

Add the folder where the binaries were installed to your user PATH by adding the following to your `.bashrc`, `.profile`, or `.bash_profile`:

```
export PATH=$PATH:/usr/local/linar/arm-linux-gnueabihf-raspbian/bin
```

At this point we must make sure that when the target (in our case the Raspberry Pi) requests a compile that the compiler names exactly match! Our experience has been that on the Raspbian target it calls for the compiler `arm-linux-gnueabihf-gcc-4.8` but the installer provided the correct version 4.8 of `gcc` but named as `arm-linux-gnueabihf-gcc`. So the only thing missing from the filename is the -4.8. So, we will create symlinks with the complete filename the target is looking for:

Inside of the `/usr/local/linaro/arm-linux-gnueabihf-raspbian/bin` create the proper symlinks:

```
$ cd /usr/local/linaro/arm-linux-gnueabihf-raspbian/bin
$ sudo ln -s /usr/local/linaro/arm-linux-gnueabihf-raspbian/bin/arm-linux-gnueabihf-gcc
/usr/local/linaro/arm-linux-gnueabihf-raspbian/bin/arm-linux-gnueabihf-gcc-4.8
$ sudo ln -s /usr/local/linaro/arm-linux-gnueabihf-raspbian/bin/arm-linux-gnueabihf-g++
/usr/local/linaro/arm-linux-gnueabihf-raspbian/bin/arm-linux-gnueabihf-g++-4.8
```

Now let's install `distcc`.

```
$ sudo port install distcc
```

1.3 Running `distcc` Server on Host Mac OS X

By default, `distcc` does not run as a server waiting for compile requests. We can start it up manually and must specify where we think the requests will be coming from:

```
$ distccd --daemon --verbose --allow '192.168.0.0/16'
```

The 0.0/16 at the end allows for any range within the 192.168.x.x subnet to connect. As of this writing, the robots are on their own subnet and this setting will allow robots to access the distcc daemon.

Alternatively, a user can specify more strict access by using the command

```
$ distccd --daemon --allow '192.168.2.0/24'
```

The 0/24 at the end allows for any range within the subnet 192.168.2.x to connect.

In order to monitor proper execution observe network activity as there should be a large spike when the target starts compiling as it is sharing files. On Mac, simply open Activity Monitor and switch to the network view.

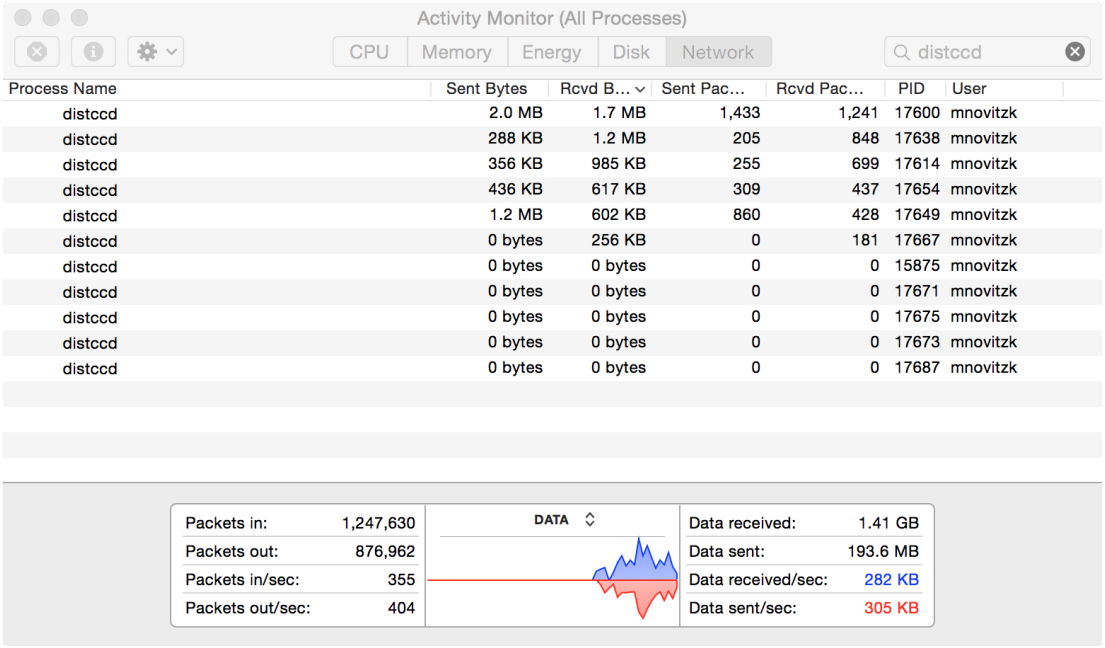

Figure 1: Mac OS X's Activity Monitor App. By switching to the 'Network' tab allows us to see the increased network activity due to `distcc` sending header files to the host to compile programs.

1.4 Execution on the target Raspberry Pi

Make sure to clean the build directories of an application you wish to compile with `distcc`. By cleaning the build directories we ensure that `distcc` will be the compiler of choice.
Perform creation of your application as normal and it should show up as `distcc`
If there is an error you will typically see:

```
distcc[6157] (dcc_build_somewhere) Warning: failed to distribute, running lcoally instead
```

This indicates that the daemon on the Host is either not running or malconfigured!

1.5 Monitoring Status

```
$ distccmon-text 2
```

The integer 2 means check every 2 seconds.