Compiling Mauve for Mac

Brief instructions for the lazy

1. Ensure that boost bjam is installed
2. Ensure that pkg-config is installed
3. svn co https://mauve.svn.sourceforge.net/svnroot/mauve/build_scripts
4. cd build_scripts
5. ./build_osx.sh

The resulting universal binaries will be located in
osx_build/mauveAligner/src

Detailed build instructions

Phase 1. Get a working compiler

Building Mauve for Mac is tricky. The aligners should be compiled for all four hardware architectures (ppc, ppc64, x86, x86_64), and should be parallel to take advantage of Apple's many dual-CPU and dual-core configurations. Unfortunately, Apple does not currently ship a compiler capable of this feat, although a gcc-4.2 alpha is currently available to apple developers for OS X 10.5.

The strategy to build Mauve for Mac has been to build a custom gcc-4.2.1 using the gcc universal build script created by ATT's R development team. The script and instructions are here:

http://r.research.att.com/tools/

There's a caveat: Apple does not support static linking, and the gcc support libraries (libstdc++, libgomp, etc) come as dynamic libraries by default. If left as is, the dynamic libraries get used and when mauveAligner and progressiveMauve are copied to another machine they fail because the requisite dynamic libraries are missing. To work around the problem, the dynamic libraries can be deleted to force static linking. It's not a particularly elegant solution but it works for me. Usually this amounts to running as root:

rm /usr/local/lib/libgomp*.dylib

rm /usr/local/lib/libstdc++*.dylib

rm /usr/local/lib/libssp*.dylib

Of course, you might prefer to move these somewhere safe instead of deleting them outright.

Phase 2. Build the aligners

Once a universal gcc has been built and installed, it's necessary to set CFLAGS and CXXFLAGS appropriately:

export CFLAGS="-O3 -g -mmacosx-version-min=10.3 -isysroot /Developer/SDKs/MacOSX10.4u.sdk -arch i386 -arch ppc -arch x86_64 -arch ppc64"

export CXXFLAGS="-O3 -g -mmacosx-version-min=10.3 -isysroot /Developer/SDKs/MacOSX10.4u.sdk -arch i386 -arch ppc -arch x86_64 -arch ppc64"

Note that CFLAGS and CXXFLAGS must be empty in order to build the universal gcc using the script linked above. It may be desirable to set these variables inside $HOME/.profile so they get configured automatically on login. Other important environment variables and their settings may be:

export PKG_CONFIG_PATH="$HOME/lib/pkgconfig"

export LD_LIBRARY_PATH="$HOME/lib"

export SVN_EDITOR=vi

export PATH="/usr/local/bin:$HOME/bin:$PATH"

At this point, the mauveAligner source and libraries can be built with the usual procedure. When running ./configure, it is necessary to add both --disable-shared and --disable-dependency-tracking for an OS

When coming to the part of building mauveAligner and progressiveMauve, the build may fail, complaining that the GOMP functions can't be found. One workaround is to copy the link command and add the path to the static libgomp library. For example:

g++ -DCOMMAND_LINE -O3 -g -mmacosx-version-min=10.3 -isysroot /Developer/SDKs/MacOSX10.4u.sdk -arch ppc -arch i386 -arch ppc64 -arch x86_64 -o progressiveMauve progressiveMauve.o UniqueMatchFinder.o -fopenmp -L/Users/mugsy/lib -lMems-1.5 -lboost_filesystem-s -lboost_program_options-s -lpthread -lClustalW-1.0 -lGenome-1.3 -lMUSCLE-3.6 /usr/local/lib/libgomp.a

Support for Mac OS X 10.3:

Unfortunately, the procedure described above won't support users of PowerPC macs running OS X 10.3. To do that, it's necessary to run a completely separate build, setting

CFLAGS="-O3 -g -mmacosx-version-min=10.3 -isysroot /Developer/SDKs/MacOSX10.3.9.sdk -arch ppc"

CXXFLAGS="-O3 -g -mmacosx-version-min=10.3 -isysroot /Developer/SDKs/MacOSX10.3.9.sdk -arch ppc"

I typically run the separate build in a separate user account. Finally, the ppc-only binaries for 10.3.9 need to be joined with the other binaries using apple's lipo tool.

Phase 3. Build the Java GUI and make a disk image

This part of the build requires a Java Development Kit and Apache Ant to be installed. With that done, copy the freshly build mauveAligner and progressiveMauve binaries into the mauve/osx subdirectory and strip them of debugging symbols:

strip osx/mauveAligner

strip osx/progressiveMauve

That reduces the distributable binary size from >100Mb to less than 10Mb. Then run:

ant macdist