FLOWER Version 06 (flr06)

**PNNL-SA-120800**

**PNNL-18428, Rev. 4**

FLOWER Build Guide

**June 2017**

**DS Curtis**

LMS Curtis

Contents

[Introduction 2](#_Toc484790455)

[Requirements 3](#_Toc484790456)

[Hardware 3](#_Toc484790457)

[Operating System 3](#_Toc484790458)

[Developer Environment 3](#_Toc484790459)

[ RHEL/CentOS 64-bit development environment 3](#_Toc484790460)

[ GNU C++11 or greater 3](#_Toc484790461)

[ Boost libraries 4](#_Toc484790462)

[ PCAP development and runtime libraries 5](#_Toc484790463)

[Building FLOWER 6](#_Toc484790464)

[Testing FLOWER 7](#_Toc484790465)

[Building RPMS 9](#_Toc484790466)

[Appendix A: GCC Build Script 11](#_Toc484790467)

[Appendix B: Boost Build 12](#_Toc484790468)

# Introduction

The FLOWER (FLOW analyzER) software was developed at PNNL and is intended for deployment on application sensors.

This guide is intended to assist an experienced developer to build and install the FLOWER software to deploy on a FLOWER appliance sensor. The instructions are written in a format that assumes a basic knowledge of Linux and C/C++ development. Short instructions are provided for each step along with command line or script references to complete the build. If you have any questions about the build, please contact FLOWER support at flower-support@pnnl.gov.

The FLOWER application is designed to start at boot time as a daemon process to summarize network flows.

FLOWER can also be used to read packets from a network interface or from packet capture ([pcap](https://en.wikipedia.org/wiki/Pcap)) files that can be processed with the [libpcap](http://www.tcpdump.org/) system library. See the FLOWER\_Ops\_Guide document for more detailed information.

# Requirements

## Hardware

The FLOWER application requires a 64-bit Intel based system with dual quad-core processors with a minimum of 4 GB of memory and 20 GB of disk.

The system requires a 1 Gigabit CAT5 or better Ethernet interface. For testing, it is recommended to have an additional 1 Gigabit CAT 5 or better Ethernet interface that is connected to a dedicated hub or switch. The hub/switch is needed to give the Ethernet interface an active link that can be used in isolation for testing. FLOWER has been tested and verified on an Intel XL710 40 Gigabit QSFP+ interface without any problems.

## Operating System

The FLOWER application can be built and installed on Red Hat Enterprise Linux or CentOS 5.2 or newer.

*It is recommended that you build and install on RHEL/CentOS 7.x*.

The Operating System can be installed to run as 32-bit only, 32-bit/64-bit mixed, or 64-bit only mode. You must build on a system that runs the same mode as the system you will be running the application on.

*It is recommended that you build only 64-bit mode. The previous modes are for legacy systems only.*

## Developer Environment

The FLOWER application requires a Linux development environment, a C++ compiler and Boost libraries. It is required to use at least the versions listed below or better.

### Operating System (RHEL/CentOS 64-bit) development environment

To ensure that you have all the necessary development tools, install the standard development environment using the following command:

# yum group install “Development Tools”

### PCAP development and runtime libraries

The PCAP development and runtime libraries are required to build FLOWER. To install them on an RHEL/CentOS system, run the following commands:

# yum install –y libpcap-devel libpcap

### GNU C++11 or greater

[*Appendix A*](#_Appendix_A:_GCC) contains a script to download and build GCC. It should work in most environments to simply copy, paste and run.

Download GNU C and C++ version 5.3 from <https://gcc.gnu.org/releases.html>.

**NOTE:** While newer versions are available, FLOWER has been tested extensively with version 5.3, so this is the recommended version.

**NOTE:**Do not use the default compiler with RHEL/CentOS as a more recent build is required for FLOWER to properly build*.*

Set the build parameters to include:

--enable-shared

--enable-threads=posix

--enable-clocale=gnu

--disable-bootstrap

--disable-multilib

Build the GNU GCC compiler tools and install them in /usr/local/gcc.

Build both the C and C++ executables and libraries.

### Boost libraries

[*Appendix B*](#_Appendix_B:_Boost) contains a script to download and build Boost. It should work in most environments to simply copy, paste and run.

Download Boost version 1.60 or greater from [http://www.boost.org/users/history](http://www.boost.org/users/history/).

**NOTE:** While newer versions are available, FLOWER has been tested extensively with version 1.60, so this is the recommended version.

**Important:** Ensure that you are using the GCC 5.3 compiler tools and not the built in C compiler tools. To ensure that you are using the proper compiler, do the following:

* + From the boost\_1\_60\_0 directory, run the following command:

# echo "using gcc : 5.3 : /usr/local/gcc/bin/g++ ; " >> tools/build/src/user-config.jam

* + Make sure you have the --toolset=gcc defined when you run your b2 or bjam command.
  + Set the following environmental variables:

unset SSH\_ASKPASS

export PATH=/usr/local/gcc/bin:/usr/local/bin:$PATH

export LD\_LIBRARY\_PATH=/usr/local/gcc/lib:/usr/local/gcc/lib64:/usr/local/lib:/usr/lib:$LD\_LIBRARY\_PATH

export CC=/usr/local/gcc/bin/gcc

export CXX=/usr/local/gcc/bin/g++

Set the build parameters to include:

link=shared

address-model=64

runtime-link=shared

threading=multi

Build the libraries and install them in /usr/local/boost.

# Building FLOWER

This section contains the instructions that build the FLOWER software. It assumes that all requirements in the previous section have been met, including the GNU C and C++ compiler tools and Boost are built and installing in /usr/local.

Download FLOWER 06 or greater from [https://stash.pnl.gov](https://stash.pnl.gov/)

# git clone https://$USER@stash.pnnl.gov/scm/flower/flower.git

**Important**: Ensure that the correct version of boost and gcc executables and libraries are in your path.

Set the following environmental variables:

unset SSH\_ASKPASS

export PATH=/usr/local/gcc/bin:/usr/local/bin:$PATH

export LD\_LIBRARY\_PATH=/usr/local/boost/lib:/usr/local/gcc/lib:/usr/local/gcc/lib64:/usr/local/lib:/usr/lib:$LD\_LIBRARY\_PATH

export CC=/usr/local/gcc/bin/gcc

export CXX=/usr/local/gcc/bin/g++

It is recommended that you use the included rebuild\_from\_scratch.sh script to build the FLOWER executables. Prior to running the script, edit and make the following changes:

* Change BOOST\_HOME to /usr/local/boost
* Change PREFIX to the location that you would like to install FLOWER. It is recommended that you use /opt/flower.

After you run rebuild\_from\_scratch.sh with no errors, run “make install”. This will install FLOWER in /opt/flower.

# Testing FLOWER

After FLOWER is built and installed, test the application as follows:

* Add flower to your path

# export PATH=/opt/flower/bin:$PATH

* Make sure the executable runs without error.

# flower –v

The output should show detailed version information, including the version of FLOWER, the version of GCC, boost, and pcap that FLOWER was built with and the build date.

* View the FLOWER help information.

# flower –h

This should show the FLOWER help information.

* Run the FLOWER test suite

Identify a network interface that is up to run the tests against (e.g. eth0)

Set the network interface variable for the test suite

# export FLOWER\_NIC=eth0

Run the test suite from the source code directory

# src/flower/api/test/flower.test

You should see output similar to the following:

Total Tests Files = 12

Total Test Cases = 62

Total Tests Failed = 0 \*\*

Exit with Status = 0 => PASSED

If these both work, proceed to Configuring FLOWER.

If any of the above tests do not work, the build did not complete correctly. If there are problems, potential areas to check are as follows:

* Ensure that the correct version of GCC is being used.
* Ensure that the correct version of Boost is being used.
* Ensure that the correct version of GCC was used during the build of Boost.
* Ensure that the PCAP and PCAP-devel libraries were installed during the build.

# Building RPMS

This section includes the instructions to build RPMS.

Prior to building, ensure that you have the RPM build utilities installed on your machine.

# yum install rpm-build

Create directories for RPM building in your home directory.

# mkdir -p ~/rpmbuild/{BUILD,RPMS,SOURCES,SPECS,SRPMS}

# echo '%\_topdir %(echo $HOME)/rpmbuild' > ~/.rpmmacros

* **GCC RPM**

Prepare the GCC components in preparation to package.

* + Build the GCC compiler tools in /usr/local/.
  + **Copy the directory to another location and name it gcc-5.3**
  + Create a tar archive of this directory

# tar cf gcc-5.3.tar gcc-5.3

* + Copy this file into your RPM build SOURCES directory.

# cp gcc-5.3 ~/rpmbuild/SOURCES

Copy the gcc.spec file from the flower/support/build/rpm directory into the ~/rpmbuild/SPECS directory.

# cd ~/dev/flower

# cp flower/support/build/rpm/gcc.spec ~/rpmbuild/SPECS

Run the RPM build for GCC.

# cd ~/rpmbuild/SPECS

# rpmbuild –ba gcc.spec

You should now have the RPM: ~/rpmbuild/RPMS/x86\_64/gcc-5.3-0.x86-64.rpm

* **Boost RPM**

Prepare the Boost components in preparation to package.

* + Build the Boost libraries in /usr/local/.
  + **Copy the directory to another location and name it boost-1.60**
  + Create a tar archive of this directory

# tar cf boost-1.60.tar boost-1.60

* + Copy this file into your RPM build SOURCES directory.

# cp boost-1.60.tar ~/rpmbuild/SOURCES

Copy the boost.spec file from the flower/support/build/rpm directory into the ~/rpmbuild/SPECS directory.

# cd ~/dev/flower

# cp flower/support/build/rpm/boost.spec ~/rpmbuild/SPECS

Run the RPM build for Boost.

# cd ~/rpmbuild/SPECS

# rpmbuild –ba boost.spec

You should now have the RPM: ~/rpmbuild/RPMS/x86\_64/boost-1.60-0.x86-64.rpm

* **FLOWER RPM**

Prepare the Flower components in preparation to package.

* + Build the Flower software in /opt/flower
  + Copy this directory into your RPM build SOURCES directory and rename to flower-6.0

# cp -R flower ~/rpmbuild/SOURCES/flower-6.0

* + Copy the flower support files from your flower source code into your RPM build SOURCES directory.

# cd ~/dev/flower

# cp -R flower/support/build/rpm/flower-support.tar ~/rpmbuild/SOURCES

* + Untar the support files into flower directory.

# cp -R flower/support/build/rpm/flower-support.tar ~/rpmbuild/SOURCES

# cd ~/rpmbuild/SOURCES/flower-6.0

# tar xf ../flower-support.tar

# cd ..

* + Create a tar archive of this directory

# tar cf flower-6.0.tar flower-6.0

Copy the flower.spec file from the flower/support/build/rpm directory into the ~/rpmbuild/SPECS directory.

# cd ~/dev/flower

# cp flower/support/build/rpm/flower.spec ~/rpmbuild/SPECS

Run the RPM build for FLOWER.

# cd ~/rpmbuild/SPECS

# rpmbuild –ba flower.spec

You should now have the RPM: ~/rpmbuild/RPMS/x86\_64/flower-6.0-0.x86-64.rpm

# Appendix A: GCC Build Script



# Appendix B: Boost Build

# 