Running the Fermi Science Tools on Windows

Thomas E. Stephens (Innovim/GSFC) and the Fermi Science Support Center Team
`fermi.gsfc.nasa.gov/ssc`

Abstract

The Fermi Science Tools, publicly released software for performing analysis on science data from the Fermi Gamma-ray Space Telescope, have been available and supported on Linux and Mac since launch. Running the tools on a Windows based host has always required a virtual machine running Linux. New technologies, such as Docker and the Windows Subsystem for Linux has made it possible to use these tools in a more native like environment.

In this poster we look at three different ways to run the Fermi Science Tools on Windows: via a VM, a Docker container, and using the Windows Subsystem for Linux. We present the steps necessary to install the tools, any issues or problems that exist, and benchmark the various installations. While not yet officially supported by the Fermi Science Support Center, these Windows installations are checked by staff when new releases are made.

Introduction

The Fermi Science Tools, now distributed as the FermiTools (see poster P4.3) are developed collaboratively between the Fermi instrument teams and the Fermi Science Support Center (FSSC). Development of these tools began before launch and the tools have been available since the very first public data release in 2009.

For a time, the Fermi Large Area Telescope (LAT) collaboration, maintained a natively compiled version of the tools for Windows, this version regularly had issues and was never publicly released or supported. It was discontinued, even for use within the LAT collaboration, in 2014.

While it has always been possible to run the tools on Windows using a virtual machine with a Linux guest operating system, this is a somewhat "heavy" solution. With the recent development of both Docker and the Windows Subsystem for Linux, it is possible for more lightweight option that allows Windows use to use the already supported Linux versions of the FermiTools in a more native fashion.

In this poster we discuss the installation and use of the FermiTools in these three different scenarios, look at some simple performance benchmarks, and discuss and issues or caveats associated with their use.

Software Installation

In a Virtual Machine

Using a virtual machine to run the tools has been available since launch. For this study, we used two different VMs both running under Oracle's Virtual Box system. One ran the Ubuntu 18.04 OS which is also run in the Windows Subsystem for Linux test and the other was running Scientific Linux 7.5, the base OS for the Docker container version of the tools.

Of the three methods a virtual machine installation is the most straight forward but also the most resource intensive, requiring the most disk space to hold the virtual machine image. Installation consists of five steps:

1. Create VM and install the guest OS
2. install a C/C++ compiler
3. install anaconda/miniconda (see poster P4.3)
4. Install the Fermi tools
5. within the FermiTools conda environment, install pyds9 via pip.

At this point the tools are ready to use.

As an optional, although desired step, on can set up a shared directory between the guest and host operating systems to store the data being analyzed. While not necessary, if Windows is your primary OS where your other tools reside, this may be a desirable configuration.

In a Docker Container

Docker provides a lighter-weight alternative to installing a full VM for running software that has been bundled into an appropriate Docker container. Since we are running Linux software on a Windows kernel, this is not as light-weight as it could be and is essentially a transparent way to run VM without having to set it up manually.

Beginning with the 2018 Fermi Summer School, the FSSC has begun providing a prebuilt Docker container with the current version of the FermiTools. Based on Scientific Linux 7, this container is available in the fssc/fermibottle repository on Docker Hub.

For this installation method, no setup within the container is needed. Simply run the container and attach to it and the tools are ready to go. The one additional installation step is to install an Xwindows system on the Windows host. This is only needed if you intend to use the graphics capabilities of the tools (e.g. plotting, gui interfaces).

The one caveat with this method is that Docker for Windows, the default Docker system, is only available for the Professional (and Enterprise) version of Windows. If you are running Window Home, you have to install the older Docker Toolkit.

In Windows Subsystem for Linux

While potentially the most seamless and "native" feeling of the installation options. using the Windows Subsystem for Linux (WSL) requires to most initial setup. It has an advantage over Docker, however, that it is available on any Windows OS, not just the Professional versions.

To install and use the FermiTools in a WSL environment requires the following steps:

1. Activate the WSL feature in your OS
2. Install a Linux distribution from the Windows store (we tested Ubuntu 18.04)
3. Install an XWindows server if desired (as for Docker)
4. Within the Linux environment
   a) Install a C/C++ compiler
   b) install anaconda/miniconda (see poster P4.3)
   c) Install the Fermi tools
   d) within the FermiTools conda environment, install pyds9 via pip.

Benchmarks

All of the benchmarking tests were run on the same system, a Dell 7250 Workstation laptop with an Intel Xeon E3-1505M v6 CPU (4 Hyper-threaded cores, 3.0 GHz) and 32 GB of RAM.

For testing, we used a set of test scripts that run the FermiTools unit tests and implement the binned and unbinned likelihood analysis from the Fermi Analysis Threads (https://fermi.gsfc.nasa.gov/ssc/data/analysis/scitools/). These tests both ensure that all the functionality is working and exercise the tools in standard analysis scenarios.

Figures 1, 2, & 3 (at right) show the timing results for the three different tests on the four different systems. Each test was run 10 times on each system.

While tests were typically run when the system was not being used for much else, and only consumed resources on 1 or 2 of the system's 8 virtual cores, some of the scatter seen is due to load from other processes on the machine. However, since this will often be the case when analyses are run with the tools, this is not really an issue. The narrower box plots for some tests are often due to those tests running overnight when the system wasn't being used for anything else.

We notice a few things right away. First the general trend is the same for all three tests. In fact, with the exception of the AGN analysis with binned likelihood, the relative ranking of the four systems is the same.

The Scientific Linux VM is the slowest of all the systems. While the cause may be that the particular VM used was not a fresh install of the OS but had been used regularly for work, the fact that the Docker container built on the same OS was the second slowest in 2 of 3 tests seems to indicate an issue with the OS itself.

The Windows Subsystem for Linux installation running Ubuntu 18.04 was by far and away the fastest of the installations across all the tests, being 10-20% faster than the SL VM depending on the test. Since the Ubuntu VM was faster than the SL VM at least some of that speed up is due to the OS but in every test, the WSL Ubuntu installation out-performed the VM installation by 7-12%. Similarly, the Docker installation always outperformed the VM installation of Scientific Linux although not by quite the same margin (only 3-9%).

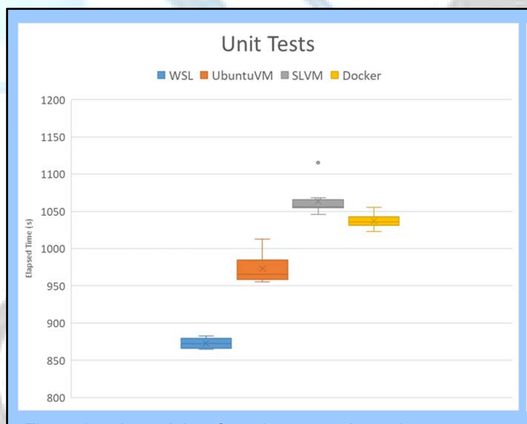

Figure 1 – elapsed time for unit tests on the various systems

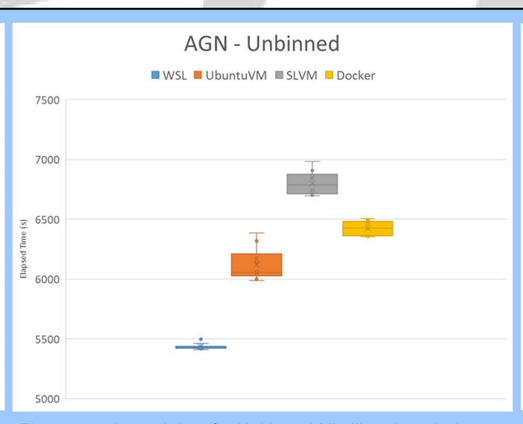

Figure 2 – elapsed time for Unbinned Likelihood analysis test on the various systems

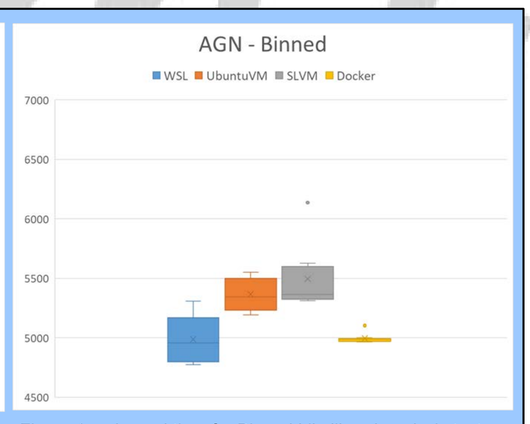

Figure 1 – elapsed time for Binned Likelihood analysis test on the various systems

Final Thoughts

The use of conda for distribution of the FermiTools greatly eases the problem of using them for scientific analysis on Windows. Because the binaries are precompiled and distributed with all of their required dependencies, users do not have to worry about configuration and compliation.

This new distribution method, combined with technology advances that provide more and easier ways to run and use Linux software on Windows, now makes that operating system a viable platform for Fermi scientific analysis without only a little bit of extra set up.

Of the methods tested, Microsoft's Windows Subsystem for Linux, while requiring the most effort to set up and install, provides the fastest processing while the Docker container requires the least user setup after getting the hosting environment configured.

Issues, Pros, and Cons

Virtual Machine

The main advantage of the virtual machine installation is its familiarity and direct correspondence to a traditional installation on a native Linux host. Once the VM is set up and running, all the tools, instructions, and help for the FermiTools apply since you are effectively working in their native environment. No additional work is required to use the tools or run an analysis.

The main downside is resource usage. Compared to the other options, the virtual machine installation is the "heaviest". It requires the most disk space for the virtual disks and there is a larger overhead for the virtual machine manager. Anecdotally (I didn't make hard measurements but was constantly monitoring CPU usage), the tests running in the VM used 3-5% more CPU than the same test running in Docker or WSL.

While these tests were done using Oracle's VirtualBox hypervisor, we expect similar results from using other hypervisors such as Hyper-V (used by the Docker container, or VMWare.

Docker Container

The first time running the tests in the Docker environment, the Binned Likelihood test failed due to lack of memory. By default, Docker gives the virtual machines it creates a 2GB memory limit. This was not enough to run the test. In order to analyze larger datasets with this installation method, you need to set the memory limit in Docker to something higher. The benchmarks presented were run with an 8GB memory limit (the same as used by the Virtual Machine installations) and had no issues. Further investigation should be done to quantify exactly how much memory is needed for different analysis scenarios.

The main downside to this method is probably the unfamiliarity with Docker and running and connecting to Docker images. Once that hurdle is overcome, this is one of the easiest methods to use since the FermiTools are already set up and configured in the downloaded Docker image allowing you to start working with them right away.

WSL

One downside of using the Windows Subsystem for Linux for the FermiTools is that it has the most complicated setup of all the methods examined. However, once set up, it is by far the easiest to use; all you have to do is launch the Linux app to get your command prompt, start the conda environment and you're off to the races. It also provides native integration with the host file system, something that takes a bit more work in the other methods.

Another issue discovered while testing this method is a problem with how WSL handles the local time zone. While the other two methods use the standard Linux time zone information, WSL creates an internal time zone based on the user's system clock. This manifested by some unit test errors that were off by an hour due to daylight savings time. This, however, affects at worst the values of times in the FITS file headers and does not affect the analysis as there were no issues with the actual analysis threads.