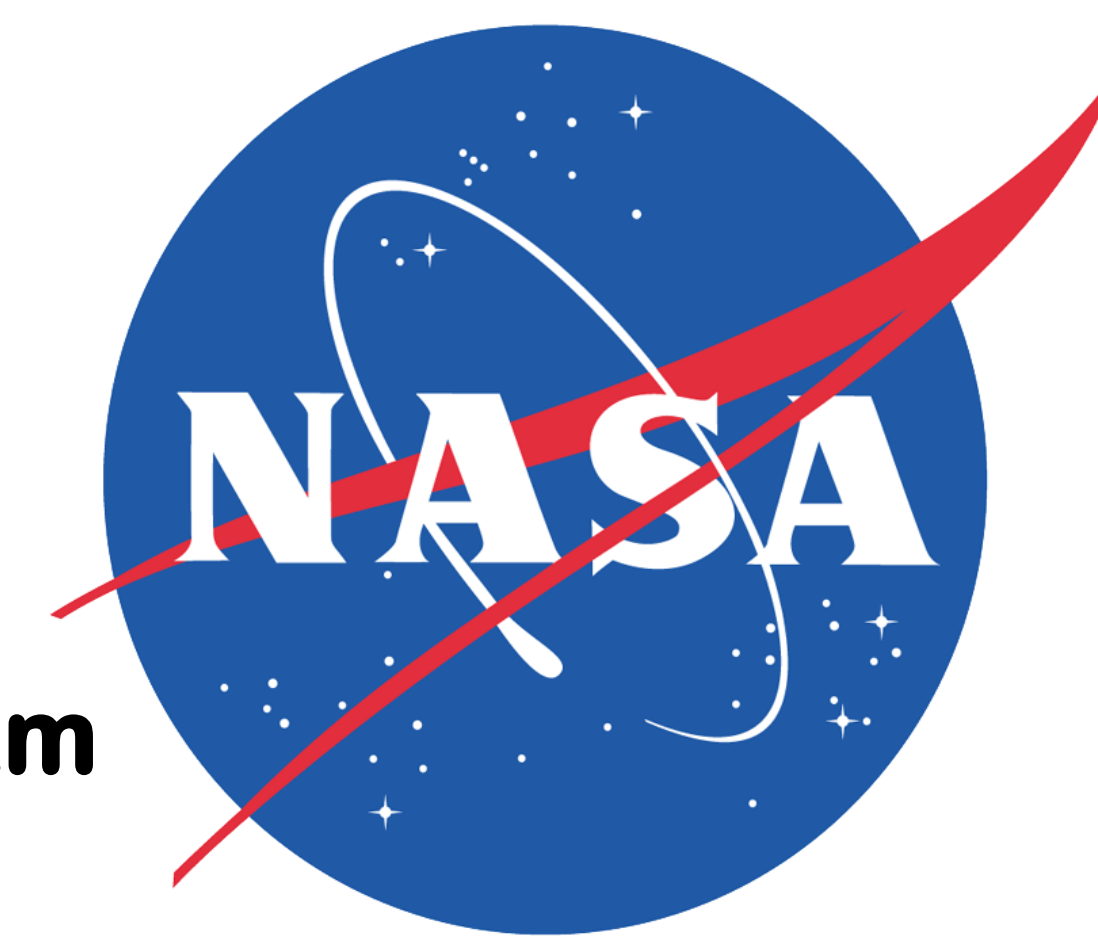


Utilizing Conda for Fermi Data Analysis Software Releases



Joe Asercion (ADNET Systems/GSFC) and the Fermi Science Support Center Team

 fermi.gsfc.nasa.gov/ssc

Abstract

Fermi Gamma-Ray Space Telescope mission provides, via the Fermi Science Support Center (FSSC), a suite of data analysis tools to assist the high energy astrophysics community in working with Fermi data. For many years these tools were distributed via both precompiled binaries and source tarball downloads on the FSSC's website. Due to the complexity of the tools and restrictions on development the downloads carried with them a large complement of third-party software which often caused package conflicts on user's machines and bloated the size of the complete analysis package. To alleviate these problems the Fermi development team has decided to update the distribution pipeline to utilize the Conda package management system. This has allowed the development team to greatly reduce the software package size, eliminate a large category of bugs which once were prevalent, and target a decrease in software update turnaround/release time. This poster will outline the process the development team took to convert our legacy codebase into a Conda compatible form and outline the lessons learned throughout this process.

Previous release system and a need for change

In the past the FSSC would generate precompiled binaries and source code tarballs that were available for download by the public. While functional, this development cycle had a number of issues which greatly extended the time between releases.

The software development team for the tools was effectively serving two customers: the internal organization users and the public users which were supported by the FSSC. Legacy design decisions prompted the creation of two separate software repositories and build systems. Changes checked in and validated at the developer level would be package and sent to the FSSC, where the code would be merged into the FSSC repository, revalidated, and reverified.

Often, due to differences in the build systems and host machines a number of issues would need to be addressed with the third party packages distributed with the tools before unit testing could begin. This greatly slowed down the development cycle and increased maintenance developer workload.

The fact that the final distributed software product contained these third party packages also made the tools prone to errors arising from incompatibilities with software already present on the user's machine. Library collisions and version incompatibilities accounted for a large percentage of technical helpdesk requests.

As part of a general overhaul of the Fermi tools development cycle the decision was made to offload management of dependencies to a package management system in order to increase ease of distribution, reduce the codebase size, and better handle the suite's complex set of software dependencies

What is Conda?

Conda is the native package management system in Anaconda and Miniconda python distributions. Developed and maintained by Anaconda, Inc (formally Continuum Analytics), it is released under the Berkely Software distribution license. Conda is open source, cross-platform, and language-agnostic.

In addition to package management functionality Conda also has environment management capability. Conda environments allow users to create self-contained installation environments for software. This prevents software installed by Conda from conflicting with software that is already installed on the user's machine or in separate conda environments. A major benefit of this functionality is that software with conflicting dependencies can coexist within the same Anaconda installation so long as they are located within different Conda environments. Unlike pip, Conda can also install different versions of python. This is incredibly useful in situations where a user needs to have access to software which requires either python2 or python3 without having to install/configure multiple versions of python in their user environment.

Conda pulls precompiled software from channels hosted on the Anaconda Cloud. While the Anaconda channel is default, users can specify what channels they would like Conda to search and the channel priority in the search order. Due to the complex set of dependencies that the Fermi tools uses this is a critical feature. The FSSC supplies a required channel list to users who wish to install the Fermi tools in order to ensure that the correct dependencies of the software are selected.

Why Conda?

Previous to the decision to move to a package management system there were ongoing discussions about replacing the version of Python2.7 that was distributed with the tools with Anaconda python due to its prevalence in the Fermi analysis community and the number of modules built into Anaconda which were already used by the tools. This proposed change along with the wide use of python for Fermi data analysis in general made the Conda Package Manager a natural choice.

Conda also allows for easy packaging and distribution of the analysis tools via the Conda Build utility and the Anaconda Cloud, respectively. The ability to take advantage of the Conda-Forge github organization's CI toolchain for long term maintenance of FSSC maintained dependencies was viewed as another major benefit.

Conda Packaging

Conda uses a built-in utility name Conda Build to package compiled binaries in a format suitable for upload to the Anaconda Cloud. To provide Conda Build with the instructions it needs to properly assemble the target software two files (at a minimum) need to be provided to the utility as part of a 'recipe': meta.yaml, which defines the build and runtime meta data of the package being built and build.sh, a bash script which directly instructs conda build in how to compile and assemble the target software.

To assemble the binaries Conda Build uses a temporary directory tree that it creates within Anaconda's directory structure to hold the source code and various build products generated during the execution of the build.sh script. Once compilation and testing (as outlined by the meta.yaml file) are complete Conda Build compresses the binary build products along with several files containing package metadata and produces a tarball. This tarball can then be uploaded to a specified Anaconda Cloud channel for distribution to the community.

Transition Process

After consolidating the Fermi tools codebase a number of changes needed to be made to the tools to accommodate the Conda Build system. Despite being compiled and packaged as a monolithic software distribution the tools themselves are maintained as separate subpackages which depend on one another. To address this issue, the build.sh script included in the Fermi tools recipe was modified to call a customized tool which had been designed specifically to properly conduct checkouts of Fermi-lat git organization software. After checkout, Conda Build is instructed to compile the Fermi tools using a specialized Scons command. Special handling for additional data files is also included within the build.sh to ensure that necessary models and reference code is included in the packaged tarball.

The meta.yaml file lists the dependencies that are required installations for the Fermi tools to run properly. All of the dependencies listed for both the build and run stages are pinned to specific versions in order to prevent possible issues caused by the third party package maintainers updating their code. The vast majority of the Fermi tools dependencies are retrieved for the Conda-Forge Anaconda channel. Conda-Forge is a github organization which contains a number of Conda recipes that are automatically tested and built using a Continuous Integration chain (AppVeyor, TravisCI, and CircleCI) to build, test, and make available for download a number of different community maintained packages. The FSSC maintains several of the Fermi tools dependencies within Conda-Forge to take advantage of the CI tools which the organization offers.

Necessary data files, and a few irregular software dependencies, are distributed by the FSSC on a separate 'fermi' Anaconda Cloud channel. The decision to separate out the data files from the primary software package allows for much more flexibility and ease in updating the data and streamlines the organization of the Fermi tools source code.

Obtaining the Fermi tools

To install the Fermi tools first a version of Anaconda Python needs to be installed. Either Anaconda or Miniconda will do, however, the FSSC recommends Miniconda as it is more lightweight. It is highly recommended that the Python2.7 version of Anaconda/Miniconda be installed. The tools Python2 at this time and while it is possible to install a different version of python in the tool's Conda environment for most users this is an unnecessary complication.

At the end of installation Conda will ask to append the its location to the front of the user's PATH in their bashrc file. Users who prefer tcsh/csh must perform an additional step to ensure that their Conda installation works properly in their shell:

```
source </path/to/conda>/etc/profile.d/conda.csh
```

must be appended to their ~/.cshrc or ~/.tcshrc file. </path/to/conda> is replaced with the path to the top level directory of the Anaconda installation (typically named something like "Miniconda2"). This is because, while Anaconda natively supports the BASH shell, it does not yet automatically complete setup for csh/tcsh.

After completing installation of the tools, a user need only run the command

```
conda create --name fermi -c conda-forge -c fermi fermi tools
```

To create a Conda environment named 'fermi' and install the latest release version of the Fermi tools into it (currently v1.0.0). Once this process is complete, the user needs only to activate the environment with the appropriate command for their shell:

Bash	Tcsh/csh
source activate fermi	conda activate fermi

After activation the user will have access to the full suite of Fermi tools installed in the 'fermi' environment.

Result

The Fermi tools is currently available to public download from the Fermi anaconda channel (for instructions on how to install the Fermi tools see the section 'Obtaining the Fermi tools').

Size Reduction in Primary Distribution

	ScienceTools-v11r5p3-fssc-20180124	Fermi tools (Tools + Data)
Mac Binary	1.44 GB (Darwin 16.7 Binary)	78.4 MB + 497.8 MB
Linux Binary	1.94 GB (Scientific Linux 7 Binary)	156.5 MB + 497.8 MB

As noted above, stripping third party dependencies from the Fermi data analysis suite substantially reduced the size of the software package that is directly managed and distributed by the FSSC.


While the dependencies must still be managed directly and version compatibility must still be monitored/tested the use of the Conda package management system simplifies this process and allows developers to focus more on the core tools.

For More Information

Fermi Science Support Center: 

Conda Documentation: 

Fermi tools Wiki: 

Conda Build Documentation: 

Fermi Anaconda Channel: 

Conda-Forge: 