IST FP6-004779

PYPY

Researching a Highly Flexible and Modular Language Platform and Implementing it by Leveraging the Open Source Python Language and Community

STREP

IST Priority 2

A Release With All Optimizations and Runtime Features, a Build- and Configuration Tool Including Documentation About the PyPy Codebase and Customization Possibilities

Due date of deliverable: March 2007

Actual Submission date: XXX insert submission date here I

Start date of Project: 1st December 2005 Duration: 2 years

Lead Contractor of this WP: merlinux

Authors: Carl Friedrich Bolz, Alexandre Fayolle, XXX

Revision: Draft

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)

Dissemination Level: PU (Public)

Revision History

Date	Name	Reason of Change
2007-01-22	Alexandre Fayolle	Describe Debian Packaging of PyPy
2007-02-01	Carl Friedrich Bolz	Add executive summary and abstract
2007-02-01	Carl Friedrich Bolz	Publishing of intermediate version on the web

Abstract

PyPy has extensive and powerful build and configuration facilities that were developed to deal with the flexibility PyPy provides. The configuration tools allow the easy addition of new configuration options and their dependencies, as well as making it easy to access the option's value and to expose the option to the user. The build enable the user to build a custom version of the PyPy interpreter locally and on remote machines.

In addition we describe the PyPy Debian-packaging effort.

Contents

1 Executive Summary

One of PyPy's foremost goals is to achieve flexibility at all levels. This flexibility gives a lot of freedom to the users and developers. On the other hand it becomes necessary to deal with the abundance of options that the flexibility induces in the various parts of PyPy. Most parts of PyPy provide various options that can be changed to adapt the part to a certain situation or a specific requirement. Some of these options need to be exposed to the end-user of PyPy so that he can adapt PyPy to his wishes. There also exist dependencies and conflicts between some of the options, since not all combinations of values make sense. To deal with this situation various tools that help building customized PyPy versions have been developed, as well as a comprehensive configuration framework that helps dealing with these problems.

The build tools are designed to ease the work of developers and users alike. It is easy for PyPy developers to specify the steps neccessary to translate PyPy and the dependencies of these steps. For the user of PyPy it is easy to specify which special features they want their translated PyPy version to have. In addition to the default build tool which builds the requested PyPy on the machine it is being run on, we developed a version that makes it possible to request a PyPy translation from a set of servers that are available for such a task. This tool makes it possible for interested people in the PyPy community to donate machine time to PyPy by making their computer available for such builds; on the other hand it lowers the barriers of entry to PyPy for interested individuals, since translating a custom PyPy version becomes very easy and does not need much (XXX 'any' when using the web interface) setup. Furthermore this automated distribution is also very interesting for PyPy developers since it becomes easy to do a large set of translations with, for example, lots of different options.

The build tools work closely together with the configuration framework to distribute the user's choices during the translation process. The configuration framework also makes it easy for PyPy developers to add new options when needed, expose them to the user, specify their dependencies and conflicts and provide documentation for the various choices.

For users who just want to use the features of the PyPy and not bother with translating themselves, the project is also providing prebuilt versions of the interpreter targeted for the Debian GNU/Linux distribution. This will also ease the task of software developers taking advantage of these features, by providing a very easy installation path of the interpreter for the people who want to try out their code, and not even remotely deal with PyPy translation.

2 Introduction

2.1 Purpose of this Document

This document describes the design of PyPy's configuration framework, PyPy's build tools and creation of Debian packages for PyPy.

2.2 Scope of this Document

While this document describes part of translation infrastucture, it does not desrcibe the translation process itself. For information about this, look at the the report "Compiling Dynamic Language Implementations" (D05.1).

2.3 Related Documents

- (D02.1) Development tools and website

3 Combined PyPy codebase

- link to walk through the code base in D14

3.1 Configuration System

3.2 Refactorings to PyPy to use the config system

3.3 interaction of translation aspects

4 Support for Building PyPy

4.1 translate.py

4.2 high-level options

4.3 buildtool

4.4 user interfaces?

5 Debian Packaging

5.1 Goals of providing Debian packages

The Debian Project is an association of individuals who have made common cause to create a free operating system. This operating system is called Debian GNU/Linux, or simply Debian for short. The emphasis put on Free Software in the Debian project has drawn a large number of highly skilled open source developpers who contribute to the project, making it the most comprehensive general-purpose GNU/Linux distribution. There are currently more than XXX Debian Developpers, and many more occasional contributors to the project, who are collaboratively maintaining more than XXX free software packages. This comprehensiveness has in turned attracted many users, both individuals and organisations (see the debian users web page for a non exhaustive list of organisations using Debian).

A Debian package is a binary file, containing all the required data and meta-data to install some software. The software is provided in a compiled form (though it is possible to download a so called "source package" with the raw source code for closer examination), eliminating the need for the user to compile the code locally, which can be unpractical, if the number of build dependencies is high, or if the compiling requires a large amount of system resources. The package meta-data include some critical dependency information, which allow the user to install the software and all the associated required packages in one single operation.

Packages which are part of the Debian project gain benefit from the Debian project infrastructure. This includes a bug tracking system, which, although it is mostly meant to communicate about issues in the packages themselves, is also widely used by users as a proxy to a program's author's own bug tracker. This makes sense since the maintainer of a Debian package is generally is close contact with the author of the program he has packaged, and will sometimes be able to patch the program to fix bugs, or to produce a fixed version of the program using patches available from the author's source tracker. Another important feature of the Debian infrastructure is the autobuilders : Debian supports 11 hardware architectures, with the appropriate automated package compiling machines for each architecture. Software packaged for Debian can potentially, with the help of developpers who dedicate part of their time to supporting these architecture, get ported to hardware platforms to which the author does not have access, thus enlarging its user base as well as its quality, since porting often involves fixing platform-specific bugs.

Since the Debian project is a Free Software project in itself, it has been used as a basis for other distributions, which use Debian as a basis, and enhance the distribution by providing new packages, or specific configurations of packages. Examples of such distributions include DeMuDi which was produced by the EU funded AGNULA project, skolelinux, and Ubuntu. Providing a Debian package means that the software can easily be installed on computers running these distributions, too.

All in all, targetting Debian means that PyPy will get a very high visibility among a public of highly skilled open source developpers, and that it will become available to a very large user base. Programs taking advantage of the unique PyPy functionnalities such as massive parallelism with stackless, or logic programming, will be easy to deploy thanks to the availability of the package.

5.2 Provided Packages

Packaging a program for Debian often means providing several binary packages, which is also known as "splitting" the program. This is necessary because of technical reasons and end user needs. For instance, two different packages cannot contain the same file (or else, they need to conflict with each other, and cannot therefore be installed simultaneously). Another example of required split is development packages: some users are only interested in using the program in a production context and do not want to install development tools on their computers. It is therefore necessary to provide a separate development package for developers which will have the required development tools and libraries in its dependency set. Providing multiple binary packages can also be a mean of providing different flavours of a program, compiled with different configuration settings, in order to suit different user needs.

With regard to the PyPy Debian packaging, the choice was made to split the source package in 6 packages: pypy-dev, pypy, pypy-stackless, pypy-logic, pypy-doc and pypy-lib. Another important package on which work was contributed is python-codespeak-lib. This is the Debian package for the pylib, which was developed as part of WP2. The work consisted in updating the existing package to suit the changes which where applied between the packaged snapshot and the 1.0 release of pylib.

5.2.1 pypy-dev

This package provides the interpreted version of PyPy, which requires a python interpreter to run. It is also meant to be used by people wanting to translate their own version of PyPy, or to compile programs written in rpython. It depends on the pypy-lib and python-codespeak-lib

packages, and on various third-party packages required for the translation process to succeed.

It may even be useful for core PyPy developers working from the source subversion repository to install this package, since it provides an easy way to install all the dependencies required to work on PyPy on a Debian system.

5.2.2 pypy, pypy-stackless, pypy-logic

These packages provide the PyPy interpreter compiled using the C backend. The pypy package provides a default interpreter with no special additional functionality, and the pypy-stackless and pypy-logic packages provide respectively access to the stackless functionalities and the logic object space. All three packages depend on the pypy-lib package.

They are meant to be installed by people who want to use the pypy interpreter, but cannot translate it themselves because they do not have access to a computer which is powerful enough to run the translation (or do not want to install the neccessary tools). They will also be used indirectly by people who want to install another program packaged for Debian which requires one of these interpreters to work.

5.2.3 pypy-lib

This package provides a version of the standard python library suitable for use with the PyPy interpreter, both interpreted and translated.

This package is here to provide files required by the pypy-dev, pypy, pypy-stackless and pypy-logic packages.

5.2.4 pypy-doc

This package contains the documentation in raw text and HTML for the PyPy project. As such, it is strongly recommended to install it when installing the other packages, and especially the pypy-dev package.

5.2.5 python-codespeak-lib

This package contains the pylib and its documentation. Installing it makes some high level commands available, such as the py.test test runner. It is also a dependency of pypy-dev.

6 Glossary of Abbreviations

The following abbreviations may be used within this document:

6.1 Technical Abbreviations:

AST	Abstract Syntax Tree
CPython	The standard Python interpreter written in C. Generally known as "Python". Available from www.python.org.
codespeak	The name of the machine where the PyPy project is hosted.
CCLP	Concurrent Constraint Logic Programming.
CPS	Continuation-Passing Style.
CSP	Constraint Satisfaction Problem.
docutils	The Python documentation utilities.
F/OSS	Free and Open Source Software
GC	Garbage collector.
GenC backend	The backend for the PyPy translation toolsuite that generates C code.
GenLLVM backend	The backend for the PyPy translation toolsuite that generates LLVM code.
Graphviz	Graph visualisation software from AT&T.
Jython	A version of Python written in Java.
LLVM	Low Level Virtual Machine - a compiler infrastructure available from University of Illinois at Urbana-Champaign
LOC	Lines of code.
Object Space	A library providing objects and operations between them, available to the bytecode interpreter via a well-defined API.
Pygame	A Python extension library that wraps the Simple Direct-media Library - a cross-platform multimedia library designed to provide fast access to the graphics framebuffer and audio device.
pypy-c	The PyPy Standard Interpreter, translated to C and then compiled to a binary executable program
ReST	reStructuredText, the plaintext markup system used by docutils.
RPython	Restricted Python; a less dynamic subset of Python in which PyPy is written.
Standard Interpreter	The subsystem of PyPy which implements the Python language. It is divided in two components: the bytecode interpreter, and the standard object space.
Standard Object Space	An object space which implements creation, access and modification of regular Python application level objects.
VM	Virtual Machine.

6.2 Partner Acronyms:

DFKI	Deutsches Forschungszentrum für künstliche Intelligenz
HHU	Heinrich Heine Universität Düsseldorf
Strakt	AB Strakt

Logilab	Logilab
CM	Change Maker
mer	merlinux GmbH
tis	Tismerysoft GmbH
Impara	Impara GmbH

References

(D02.1) *Configuration and Maintenance of Development Tools and Website*, PyPy EU-Report, 2007

(D05.1) *Compiling Dynamic Language Implementations*, PyPy EU-Report, 2005