IST FP6-004779

PYPY

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

STREP

IST Priority 2

D05.2: Release a compiled, self-contained modular static version of PyPy

Due date of deliverable: August 2005

Actual Submission date: 23th December 2005

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

Lead Contractor of this WP: HHU

Authors: Jacob Hallén (AB Strakt), Anders Lehmann (DFKI GmbH)

Revision: final

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

Dissemination Level: PU (Public)

Abstract

This document describes the compiled, self-contained, modular implementation of Python as available in the 0.7 release of the PyPy project, made on Sun 28 Aug 2005.

Contents

1 Executive Summary

The PyPy interpreter for the Python programming language is implemented in a restricted subset of Python called RPython. This makes it possible to either run it in a Python interpreter or to generate a self-supporting executable. The advantage of the latter option is primarily that we no longer pay the performance penalty of running our Python interpreter inside another Python interpreter. It also allows us to apply a number of optimisations that will further improve the speed of execution.

In this stage of the project we have implemented a number of steps that do the translation of the interpreter that is the product of workpackage 4 into a stand-alone executable. Components involved in this include a Flow Object Space, an Annotator, the RTyper and a target specific backend. At this stage of the project we have implemented two different backends; one producing C-code and one producing code for the Low Level Virtual Machine (LLVM). The output of the former backend is compiled into machine code for a specific processor architecture using a C compiler. The latter goes through a further translation step before being compiled for a specific process architecture using the LLVM compiler toolchain or a C compiler.

The compiled C backend turned out to be 200 times slower than CPython and a factor of 10 times faster than pypy running on top of CPython. Execution speed was considerably improved shortly after the release.

2 Introduction

2.1 Purpose of this Document

This document describes the 0.7 release of PyPy - The first self contained version of PyPy.

2.2 Scope of this Document

This document describes the contents of release 0.7. However, it omits all details about basic interpretation, as these are covered in deliverables D04.1 and D04.2. Neither does it cover the details of type inference and translation. These aspects are covered deliverables D05.1, D05.3 and D05.4.

2.3 Related Documents

This document has been prepared in conjunction with other tasks in work packages 3, 4 and 5. Before reading this document, a close look at the following deliverables is recommended.

- D04.2 Complete Python implementation running on top of CPython

3 Achieved Goals

The second public release of the PyPy interpreter was released on Sun 28 Aug 2005, with version number 0.7.0.

The release is available at:

> http://codespeak.net/download/pypy/pypy-0.7.0.tar.bz2 and
> http://codespeak.net/download/pypy/pypy-0.7.0.tar.gz and
> http://codespeak.net/download/pypy/pypy-0.7.0.zip respectively.

3.1 WP3 (Synchronisation with Standard Python)

Updated PyPy to Python2.4.

3.2 WP4 PyPy Core

See D04.2 for details.

3.3 WP5 Translation

Task 1 (Completed)

> *At the core of this task lies the research and implementation of full "abstract interpretation" by reusing the unmodified PyPy interpreter with a special object space. Create a code analysis tool for a subset of the Python language (RPython). Coordinate the definition of RPython with WP04, being the implementation language for most of the core. Experiment with different ways of doing Abstract Interpretation. Look for an Abstract Domain suitable for type inference. Compare with published algorithms for type inference. Select and implement the most appropriate solution in our context.*

Task 2 (Mostly completed)

> *Produce a toolchain, capable of extracting bytecode of a RPython program from the core, translating it into low-level code (with C being the primary target) and compiling it. Create a build process for statically generating and running a low-level PyPy interpreter and object space. The LLVM (Low Level Virtual Machine) project is a candidate for collaboration in this context. Provide hooks into internals to alter translation aspects. Research/experiment with a Java backend.*

A toolchain was developed to enable translation of PyPy to C and LLVM. In January 2005, after the launch of the project, we came into contact with Impara, who are doing research on Squeak, a self hosted variant of Smalltalk. This has led to Impara joining the project as a partner for the purpose of doing research and experiments with translation to a high level language backend, i.e. Squeak. While this work happens after the end of phase 1, it does not block any other work. Experiments have also been done with Javascript as the target language. Some early work on a Java backend has been done, but this has been mothballed. We may decide to do some further experiments with Java after a stable API for high level language backends has been developed.

Task 3 (Completed)

In order to give a working environment to the translated RPython program, build the low-level-specific runtime components of PyPy. For the C PyPy runtime, important parts can be directly re- used from CPython. However, certain aspects such as memory management and threading models are to be implemented in a modular way. Implement a minimal C-runtime for the translated PyPy, collaborate for with CPython core developers. Integrate Memory Management, threadings models, and other aspects in a modular way into both the C-runtime and the statically generated PyPy low level code. For these translation aspects improve and build on top of solutions from other open source language projects.

A modular approach to codegeneration was developed allowing different strategies for memory management (ref counting and Boehm garbage collectors), thread models, backends (C and LLVM) and platforms (tested on Mac OS X, Linux and Windows).

4 Public Announcement of the Release

This is a copy of the release announcement:

```
pypy-0.7.0: first PyPy-generated Python Implementations
++++++++++++++++++++++++++++++++++++++++++++++++++++++++

What was once just an idea between a few people discussing
on some nested mailing list thread and in a pub became reality ...
the PyPy development team is happy to announce its first
public release of a fully translatable self contained Python
implementation.  The 0.7 release showcases the results of our
efforts in the last few months since the 0.6 preview release
which have been partially funded by the European Union:

- whole program type inference on our Python Interpreter
  implementation with full translation to two different
  machine-level targets: C and LLVM

- a translation choice of using a refcounting or Boehm
  garbage collectors

- the ability to translate with or without thread support

- very complete language-level compliancy with CPython 2.4.1

What is PyPy (about)?
+++++++++++++++++++++

PyPy is a MIT-licensed research-oriented reimplementation of
Python written in Python itself, flexible and easy to
experiment with.  It translates itself to lower level
languages.  Our goals are to target a large variety of
platforms, small and large, by providing a compilation toolsuite
that can produce custom Python versions.  Platform, Memory and
Threading models are to become aspects of the translation
process - as opposed to encoding low level details into a
language implementation itself.  Eventually, dynamic
optimization techniques - implemented as another translation
aspect - should become robust against language changes.
```

```
Note that PyPy is mainly a research and development project
and does not by itself focus on getting a production-ready
Python implementation although we do hope and expect it to
become a viable contender in that area sometime next year.


Where to start?
+++++++++++++++

Getting started:    http://codespeak.net/pypy/dist/pypy/doc/getting-started.html

PyPy Documentation: http://codespeak.net/pypy/dist/pypy/doc/

PyPy Homepage:      http://codespeak.net/pypy/

The interpreter and object model implementations shipped with
the 0.7 version can run on their own and implement the core
language features of Python as of CPython 2.4.  However, we still
do not recommend using PyPy for anything else than for education,
playing or research purposes.

Ongoing work and near term goals
++++++++++++++++++++++++++++++++

PyPy has been developed during approximately 15 coding sprints
across Europe and the US.  It continues to be a very
dynamically and incrementally evolving project with many
one-week meetings to follow.  You are invited to consider coming to
the next such meeting in Paris mid October 2005 where we intend to
plan and head for an even more intense phase of the project
involving building a JIT-Compiler and enabling unique
features not found in other Python language implementations.

PyPy has been a community effort from the start and it would
not have got that far without the coding and feedback support
from numerous people.   Please feel free to give feedback and
raise questions.

    contact points: http://codespeak.net/pypy/dist/pypy/doc/contact.html

    contributor list: http://codespeak.net/pypy/dist/pypy/doc/contributor.html

have fun,

 the pypy team, of which here is a partial snapshot
 of mainly involved persons:

    Armin Rigo, Samuele Pedroni,
    Holger Krekel, Christian Tismer,
    Carl Friedrich Bolz, Michael Hudson,
    Eric van Riet Paap, Richard Emslie,
    Anders Chrigstroem, Anders Lehmann,
    Ludovic Aubry, Adrien Di Mascio,
    Niklaus Haldimann, Jacob Hallen,
    Bea During, Laura Creighton,
    and many contributors ...
```

```
PyPy development and activities happen as an open source project
and with the support of a consortium partially funded by a two
year European Union IST research grant. Here is a list of
the full partners of that consortium:

    Heinrich-Heine University (Germany), AB Strakt (Sweden)
    merlinux GmbH (Germany), tismerysoft GmbH(Germany)
    Logilab Paris (France), DFKI GmbH (Germany)
    ChangeMaker (Sweden), Impara (Germany)
```

5 Glossary of Abbreviations

The following abbreviations may be used within this document:

5.1 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.
docutils	The Python documentation utilities.
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 Directmedia Library - a cross-platform multimedia library designed to provide fast access to the graphics framebuffer and audio device.
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.

5.2 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