PYPY

Scope

Traditionally, flexible computer languages allow to write a program in short time but the program then runs slower at the end-user. Inversely, rather static languages run faster but are tedious to write programs in. Today, the most used flexible and productive computer languages are hard to get to run fast, particularly on small devices. During its EU project phase 2004-2007 Pypy challenged this compromise between flexibility and speed. Using the platform developed in Pypy, language interpreters can now be written at a higher level of abstraction. Pypy automatically produces efficient versions of such language interpreters for hardware and virtual target environments. Concrete examples include a full Python Interpreter, whose further development is partially funded by the Google Open Source Center – Google itself being a strong user of Python. Pypy also showcases a new way of combining agile open-source development and contractual research. It has developed methods and tools that support similar projects in the future.

Advances

Traditionally, language interpreters are written in a target platform language such as C, Java or C#. One of the goals of the «all-encompassing» environments, like the .NET framework and the Java virtual machine, is to provide standardized and higher level functionality in order to support language implementers in writing language implementations. Pypy took a more ambitious approach. We defined a subset of the high-level language Python in which we implement languages as simple interpreters with few references to and dependencies on lower level details. Our translation framework then produces a concrete virtual machine for the platform of our choice by inserting appropriate lower level aspects. The result can be customized by selecting other feature and platform configurations. For example, one can rather easily get a customized Python version for embedded environments such as mobile phones or one for enabling gamers to program parts of a massive multiplayer game such as Second Life, as Pypy also provides new ways for securing and sandboxing the execution of a program.

Positioning in global context

Pypy's Python Interpreter on the one hand competes with other existing Python implementations, on the other hand there is good collaboration between the developers of these Python implementations. Pypy is widely recognized as following a more ambitious and far reaching approach. After its next release end of the year 2008 we expect main developers and companies to make use of the new Python implementation. Pypy is very well equipped to be adapted to new platforms, to be used in education and to give new answers to security questions.

Contribution to standardization and interoperability issues

Pypy does not contribute to formal standardization processes. However, Pypy's Python implementation is practically interoperable and compatible with many environments. Each night, around 12 000 automatic tests ensure that it remains robust and compatible to the mainline Python implementation.

Target users / sectors in business and society

Potential users are developers and companies who have special needs for using productive dynamic computer languages.

Pypy, as an Open Source project, does not only consist of the EU consortium but also of a growing community of interested developers, scientists and contributors around it. Since its start – before, during and after it received EU funding – Pypy has been carried forward by this community, with its own interests and contributions.

Pypy's sprint-driven development approach pioneered a new hybrid model for combining the contractual obligations of the EU Sixth Framework research funding with the Open Source culture, and the creation of a Pypy community that can sustain itself beyond the funding period. Nineteen week-long intense physical meetings («sprints») in various international locations were the key factor for successful project development and for integrating and mentoring new contributors. Pypy's hybrid mix of distributed agile and plan-driven development practices has become a research topic of its own. It may serve as a blueprint for integrating EU funded research with Open Source engineering.

Overall benefits for business and society

Pypy increases productivity of software development. It allows to bring dynamic computer languages to environments where it previously wasn't available. It provides more robust and secure execution of programs. It provides new methodologies for efficient software development. These advantages will, in the long run, render software better and more affordable.

Examples of use

Pypy's Python implementation is on the tipping point of being complete and practical enough to become used by a wider developer community. Since a few weeks it is now possible to run existing Python applications such as Django (a popular web application framework) on top of our robust and fast Interpreter. Our implementation provides means to run untrusted programs in a very trusted way. We thus expect Web application and game companies with a particular need for running user-provided programs in their environments to look into using our Python Interpreter in the near future.

Another strong use case arises in the context of targeting small devices such as mobile phones. Here Pypy can provide a Python implementation that suits memory and power consumption requirements very well – without having to manually modify the language implementation.

Our popular testing tool is used from many hundred projects and companies, independently from the rest of the project. Offering extensions, giving tutorials and teaching about development processes arise as business scenarios.

Achievements

The Pypy project produced three major exploitable results:

- **The Pypy computer language implementation platform**: Pypy supports multiple different dynamic languages, with Python, JavaScript and Prolog currently implemented. Pervasive use of metaprograming and advanced methods for translation and just-in-time optimizations provide high performance on a number of backend systems. There is support both for low level backends like C/POSIX and LLVM (Low Level Virtual Machine) as well as high level ones, including CLI/.NET, Smalltalk and JVM.
- **The flexible Python interpreter**: The most mature interpreter written in the Pypy implementation platform is a fully compliant interpreter for the Python 2.4 language. It is more flexible and open to language research and enhancements than any pre-existing implementation of Python, supporting novel features that give programmers better solutions to existing problems (security, distribution, parallelism, logic programming, etc.). The interpreter is available under the MIT Open Source License.
- **The py.test testing framework** is an advanced and flexible testing tool written in the Python programming language. Compared to other unit testing frameworks for Python, it requires much less boiler plate code, gives better control over the testing process and makes use of idioms that are closer to the Python programming philosophy. It provides cross-platform and distributed testing extensions.

All results are published under the MIT Open Source License. Read more on Pypy at http://codespeak.net/pypy/dist/pypy/doc/home.html.

title
Research a highky flexible and modular language platform and implementing it by leveraging the open source python language and community

contract number
004779

type of project
Specific Targeted Research Project

contact point
Stephan Busemann Burt
DEUTSCHES FORSCHUNGSZENTRUM FUER KUENSTLICHE INTELLIGENZ GMBH, DE
e-mail: Stephan.Busemann@dfki.de

project website and partner list
http://codespeak.net/pypy/dist/pypy/doc/

EC contribution
1 353 000 €

start date
01/12/2004

duration
28