Contribution ID: **62** Type: **not specified**

The Essentials of Stackless Python

Tuesday, 10 July 2007 10:00 (30 minutes)

This is a re-worked, actualized and improved version of
my talk at PyCon 2007. Repeating the abstract:

As a surprise for people who think they know Stackless,
we present the new Stackless implementation For PyPy,
which has led to a significant amount of new insight
about parallel programming and its possible implementations.
We will isolate the known Stackless as a special case of
a general concept.

This is a Stackless, not a PyPy talk. But the insights presented
here would not exist without PyPy's existance.

Summary

Stackless has been around for a long time now. After several
versions with different goals in mind, the basic concepts of
channels and tasklets turned out to be useful abstractions,
and since many versions, Stackless is only ported from version
to version, without fundamental changes to the principles.

As some spin-off, Armin Rigo invented Greenlets at a Stackless
sprint. They are some kind of coroutines and a bit of special
semantics. The major benefit is that Greenlets can run on
unmodified CPython.

In parallel to that, the PyPy project is in its fourth year
now, and one of its goals was Stackless integration as an option.
And of course, Stackless has been integrated into PyPy in a very
nice and elegant way, much nicer than expected. During the design
of the Stackless extension to PyPy, it turned out, that tasklets,
greenlets and coroutines are not that different in principle,
and it was possible to base all known parallel paradigms on
one simple coroutine layout, which is as minimalistic as possible.

It is a side effect of PyPy's simplicity, that even led to a
pretty new concept for Stackless, that allows all the different
switching paradigms to co-exist without interference. Users
could go further and implement their own concurrency model,
and it would neither interfere with others nor cost performance.

Today's Stackless can be seen as a special case of the more general
implementation given for PyPy. This implementation can also be taken
as a reference about how Stackless is meant to be implemented.
This reference implementation is completely written in Python.

The talk will try to isolate the crucial design decisions in Stackless from implementation details. The reduced concepts are together the essentials of Stackless.

Special emphasis is given to interactive examples, simple use-cases, and an animation that visually explains the new concept of composability.

Primary author: TISMER, Christian (tismerysoft GmbH)

Presenter: TISMER, Christian (tismerysoft GmbH)

Session Classification: Python Language and Libraries

Track Classification: Python Language and Libraries