The Nix Build Farm: A Declarative Approach to Continuous Integration

Eelco Dolstra Eelco Visser

Delft University of Technology, EWI,
Department of Software Technology

July 8, 2008

Build farms

Build farm: a set of machines that continuously builds and tests software components from a version management system, producing status reports and/or releases.

Build farm goals

- Portability testing
  - Windows, Linux distributions, Mac OS X, 32 bit vs 64 bit, ...
- Integration testing
- Test many configurations
  - Debug vs optimised version
  - Does it build with GCC 3.4 / 4.0 / 4.1...?
- Run large regression tests
- Run analysis tools
  - Code coverage, FindBugs, ...
- Produce releases
  - Source releases, RPMs, ...

Example

```
List with some elements
strategy failed
List with element of illegal type
List with element of illegal type
Empty list
[ lt-dfta-accept-tests | critical ] No productive start symbols
left in rtg
RTG(Start([]),ProdRules([]))
FAIL: dfta-accept-tests
========================================
1 of 2 tests failed
Please report to stratego-bugs@cs.uu.nl
========================================
make[4]: *** [check-TESTS] Error 1
make[4]: Leaving directory
`/tmp/nix-24398-5/svn-export/stratego-libraries/rtg/tests'
make[3]: *** [check-am] Error 2
```

Example

PHP-SAT, the PHP static analysis tool release php-sat-0.1pre286

This page provides release **php-sat-0.1pre286** of PHP-SAT, the PHP static analysis tool. It was generated automatically on 2006-11-14 22:13:35 UTC from revision 286 of the path /php-sat/trunk of its Subversion repository (the XML record of the build job is available).

Distribution

Binary archive for Microsoft Windows

- `php-sat.zip` (10642950 bytes; MD5 hash: `9ce5bb9f87a613803547cece51c1d451`)

RPM for Red Hat 9.0

- `php-sat-0.1pre286-1.i386.rpm` (145051 bytes; MD5 hash: `fcfdcd512e3c9e6e548d0bbbb0647bba`)
- `php-sat-0.1pre286-1.src.rpm` (551573 bytes; MD5 hash: `f06c9bfc1ac95041ce52ab61e7df64a9`)

This RPM requires that the following packages are also installed:

- aterm-2.4.2-1.i386.rpm
- php-front-0.1pre287-1.i386.rpm
- sdf2-bundle-2.3.4pre15345-1.i386.rpm
- strategoxt-0.17M3pre15898-1.i386.rpm

RPM for SuSE 9.0

Current build farm tools

Examples

- Mozilla Tinderbox
- CruiseControl
- AntHill
- BuildBot
- SourceForge Compile Farm

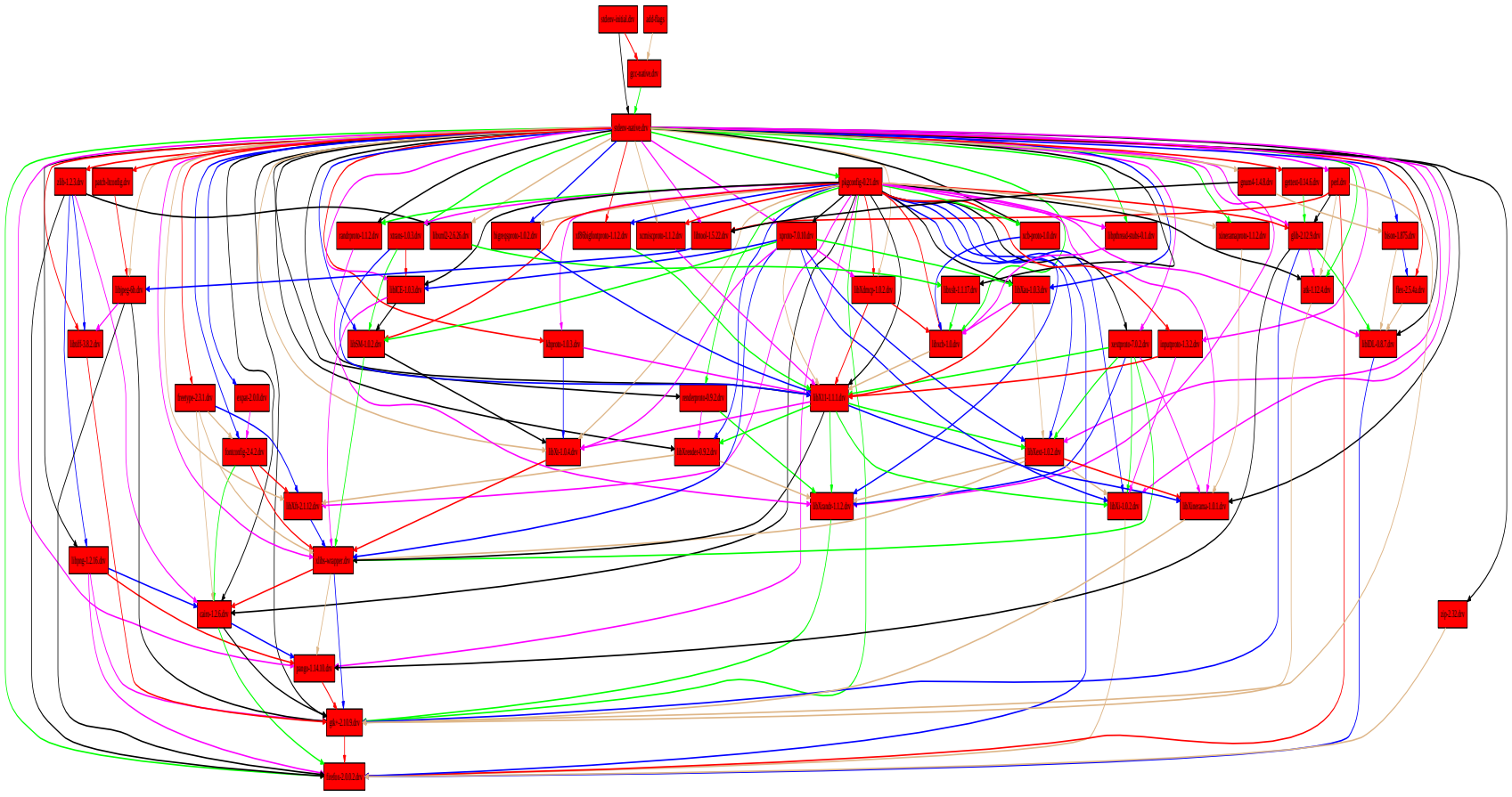
Central Problem

How do we manage the build environment?

Problem: creating the build environment

- A package typically has a lot of build time dependencies that must be distributed to each build machine
- N dependencies, M platforms
  $\Rightarrow \Theta(\ N \times M)$ effort to keep the build farm up to date
- And what if there are conflicting dependencies?

Example: build-time dependencies of Firefox



Firefox

Solution: the Nix package manager

- Package manager developed at Utrecht University, TU Delft: `http://nixos.org/`
- *Purely functional* package management:
  - Purely functional language to describe how to build packages.
  - Build results only depend on declared inputs.
  - Packages never change after they have been built.
- This is exactly what we need for a build farm:
  - Describe build jobs / dependencies in the Nix language
  - Reproducible, deterministic, automatic
  - Functions to express variability

Current status

Nix build farm is currently used to build many projects:

- Nix itself
- The Stratego/XT program transformation toolsuite: many packages
- EHC, a Haskell compiler
- More than 600 packages in the Nix Packages collection, built on several platforms

Lessons

- As a deployment tool, Nix has to be easily deployable ⇒ written in C++, Perl (not Haskell)
- Purely functional DSLs can be efficiently and easily implemented using maximal sharing (ATerms)
  See our LDTA-2008 paper
- Important for disseminating research results to make tools (e.g. Stratego/XT) easily deployable
  Continuous build / release is very useful for this

More information

- `http://nixos.org/`
- `http://buildfarm.st.ewi.tudelft.nl/`