TULIP
Continuous testing of Linux distributions upgrade

Stefane Fermigier
Laurent Godard
Agenda

- Context - EDOS WP3
- Tulip Framework
- Results and future enhancements
Context - EDOS WP3

- Testing Framework and Quality assurance portal
- QA Tools for linux distributions actors
- Accessible & easy to use
Problem to solve

Upgrading a distribution is a risky business

- Specially when using “unstable” branches
- Ex: upgrading perl on Cooker usually breaks vi (!), and sometimes urpmi, as I have personally witnessed 5 times over the last 5 years
- Users want “transparent upgrades” that work
- Even for unstable branches

Both a packaging and a tool (apt, urpmi...) problem (a repository replication / network issue too)
How to solve it?

These problems can be addressed at the theoretical level (WP2)
- Good because this is an algorithmically very complex problem

But we need also testing tools to reasonably check that things actually work for end users
- Must aim for rough tests ("smoke tests") because we can’t address the whole problem space (∼2^n combinations, where n = # of packages)
Portal V1

- At qa.edos-project.org
- Knowledge base of articles, software and other resources on testing and QA
- Reports on static package analysis (Cooker)
Next steps for Portal (V2-V3)

Create and define the testing farm

Tools to manipulate it

- Administration
  - New project
  - Add tests
  - Add reports
- Consultation
  - Reports
  - Alerts
QATR – QA Test Runner

A versatile “unit” testing framework for packages

Prototype available

Test the applications once installed
  • Setup/tear down environment
  • Unit tests
  • Doctests

QA at application/package level

Can be integrated at a higher level

Similar to the qmtest project, still pondering if it’s sensible to base our work on qmtest or not
TULIP Framework

- Testing
- Upgrades of
- Linux
- Images
- Program

Drive upgrade tests of various linux distributions to ensure both fine grained QA at the package level and testing of the standard update mechanism

Inspired by continuous integration testing frameworks (Continuum, Buildbot, Cruisecontrol)
Constraints

- Modify the system as least as possible
  - Minimize « Observer effect »
  « ... instruments that by necessity alter the state of what they measure ... » source Wikipedia (http://en.wikipedia.org/wiki/Observer_effect)

- Few hypotheses on what is present/correct in the image

- Run all the projects on a daily basis max: the testing machine has to be able to respond to periodicity minima

- Storage issues
Global architecture

Used tools
- Qemu (incremental qemu images)
- Python + pexpect + SQLAlchemy
- SSH
- Some basic console commands
- BIRT
- Sqlite, Postgresql at the end

A main runner
A project creator (local)
A database creator (local)
Prerequisites on image

- Ssh and root access allowed
- Silent/non-interactive mode of upgrade
- Python
- Size problem solved with incremental images of qemu
- Compression ratio ~ 1:4
Define a project

- Name
- Installer type
- Periodicity (daily, weekly, monthly)
- Paths (linux image, storing area)
- Ssh connection parameters

2 projects under test so far:
- Debian Etch (= testing) Gnome desktop
- Mandriva 2006 Community KDE desktop
Run a session

- Unzip qemu image of the last session
- Launch qemu image (or equivalent) (with parameters like tcp redirection or memory size)
- Connect as root through ssh
- Loop over update and upgrade commands
- Retrieve result (log file)
- Loop over tests
  - Download and run tests
  - Upload tests results
- Shutdown image
Delta Sessions

Depending on periodicity, N past images are kept
- Daily 6 images
- The Delta 1 is the current (most recent) image

Moved down in the hierarchy at each session: the new upgraded image becomes Delta 1 and the other decreased by one.

Allow more large steps in upgrading where more changes occur on mirrors at a time
Running post-install tests

- Verify that all (or targeted) packages/software are operational after the upgrade
- Binary dependancy (script using ldd), explicit QA scripts (QATR) ...

Synopsis
- All in a main loop
- Tests files are uploaded (engine if necessary)
- Tests return a list of faulty packages
- List retrieved back to the session-pilot and parsed in database

Caution: may be time-consuming
Collected informations

- Depending on installer and distribution
- Upgraded Package name
- Version
- Previous version
- Success/failure/name of the test
- Global execution time
- ...
BIRT reports

- Eclipse tool
- Create reports bound to a data source
- Graphical and list representation depending on parameters
- Output as PDF, HTML or displayed using integrated viewer
- Built-in parameters selections
Session Report

TULIP Project Activity - Session report

Session Reference: 195  Date: 28-Jul

Project Name: debian  Project Id: 11  Period: daily

Number of packages: 14  Processed Packages List for the session (delta):

- hicolor-icon-theme
- modutils
- grub
- nano
- mount
- libsysfs2
- libc6-dev
- libc6
- glibc
- util-linux
- locales
- debconf
- hicolor-icon-theme
- modutils
- grub
- libc6

Images Delta Upgrades results

<table>
<thead>
<tr>
<th>Delta Rank</th>
<th>Package name or error</th>
<th>Status</th>
<th>version</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>hicolor-icon-theme</td>
<td>ok</td>
<td>0.8-4</td>
</tr>
<tr>
<td>4</td>
<td>modutils</td>
<td>ok</td>
<td>2.4.27-0.6</td>
</tr>
<tr>
<td>4</td>
<td>grub</td>
<td>ok</td>
<td>0.97-11</td>
</tr>
<tr>
<td>4</td>
<td>nano</td>
<td>ok</td>
<td>1.3.11-3</td>
</tr>
<tr>
<td>4</td>
<td>mount</td>
<td>ok</td>
<td>2.12r-10</td>
</tr>
<tr>
<td>4</td>
<td>libsysfs2</td>
<td>ok</td>
<td>2.0-0-7</td>
</tr>
<tr>
<td>4</td>
<td>libc6-dev</td>
<td>ok</td>
<td>2.3.6-15</td>
</tr>
<tr>
<td>4</td>
<td>libc6</td>
<td>ok</td>
<td>2.3.6-15</td>
</tr>
<tr>
<td>4</td>
<td>glibc</td>
<td>ok</td>
<td>2.4.2-2</td>
</tr>
<tr>
<td>4</td>
<td>util-linux</td>
<td>ok</td>
<td>2.12r-10</td>
</tr>
<tr>
<td>4</td>
<td>locales</td>
<td>ok</td>
<td>2.3.6-15</td>
</tr>
<tr>
<td>2</td>
<td>debconf</td>
<td>ok</td>
<td>1.5-2</td>
</tr>
<tr>
<td>2</td>
<td>hicolor-icon-theme</td>
<td>ok</td>
<td>0.8-4</td>
</tr>
<tr>
<td>2</td>
<td>modutils</td>
<td>ok</td>
<td>2.4.27-0.6</td>
</tr>
<tr>
<td>2</td>
<td>libc6</td>
<td>ok</td>
<td>0.97-11</td>
</tr>
<tr>
<td>2</td>
<td>glibc</td>
<td>ok</td>
<td>2.12r-10</td>
</tr>
<tr>
<td>2</td>
<td>libc6-dev</td>
<td>ok</td>
<td>2.3.6-15</td>
</tr>
<tr>
<td>2</td>
<td>glibc</td>
<td>ok</td>
<td>2.12r-10</td>
</tr>
</tbody>
</table>

Si...
TULIP Project History Report

TULIP Project Activity - Project History

- Project Name: debian
- Project Id: 11
- Periodicity: daily
- Installer type: apt-get

Number of sessions:
- From ...: 2006-06-12 15:45:56
- To ...: 2006-07-03 08:42:40

Effective Upgrades results

- % Upgrade successfull
- Number of errors
- Installed packages
- Errors

Stefane Fermigier – Laurent Godard - contact@nuxeo.com
Some figures

- Typical daily project size: 8 Gb
  - (6 Gb base + 2 Gb incremental images zipped)
- Duration: ~ 2 hours
- Number of daily upgraded packages: 0-10
- Number on weekly basis (Delta 6): 10-50
First results

- Still under development
- No package errors detected on “stable” distributions testing
  - On upgrade
  - Regarding binary dependency checking
- Urpmi log file is too weak (more a screen dump than a log) – Hard to parse reliably
- Some connection issues regarding --curl default of urpmi
  - Use of --wget switch recently
To do

- Run on “unstable” distribution branches (Cooker, Debian Unstable)
- Integrate to portal
  - Create projects, launch manually, add or see reports ...
- Re-work urpmti analysis as log file is not reliable: first simulate then upgrade and finally compare. Is a generalization useful?
Integrate “unit” tests (QATR, others)

Other distributions and other installers
- Ready (using urpmi or apt-get) Ubuntu, Kubuntu
- yast, yum: only need a parser for installed packages and command lines
Planned enhancements

Projects
- More installation profiles (desktop/server, stable/unstable)
- Test upgrades from stable to unstable/testing

More reports
- Follow a package through time

Allow pre-update & pre-upgrade scripts
- Change sources, add new packages
- Repair a broken image

Alerts
- Email, RSS feed, Jabber, Nabaztag...
Possibilities

Collect more information
  • Hardware stress statistics

Test other categories with new projects
  • Mirrors availability/out-of-sync
  • A reference repository and verify selected mirrors are providing the same results

Even more reports
Conclusion on TULIP

- Framework for testing upgrade of various Linux images
- Incremental delta sessions increase possibilities for coverage
- Can be enhanced easily
  - More installers (and distributions)
  - More install typologies
  - More tests
  - More reports
Perspectives for Portal V3

- Alerts
- The testing farm is on the road
  - Validate collected metadata
  - Define reports and metrics
- A project manager for piloting the testing farm
  - Interactive job definition and scheduling
- A interactive report manager for adding and calling new BIRT reports
Tulip inspired by prior work by Nexedi (Umigumi/Umitester) and Caixa Magica (eqatool)

Qatr inspired by qmtest (from CodeSourcery) and the unit-testing movement (Cunningham, Beck...)

Discussion with other EDOS projects members

Contributors at Nuxeo: Laurent Godard, Stefane Fermigier, Benoit Delbosc, Tarek Ziadé, Olivier Grisel...