Use of Grid Computing for Debian Quality Assurance

Lucas Nussbaum
lucas@debian.org – lucas.nussbaum@imag.fr

Laboratoire d'Informatique de Grenoble - Projet MESCAL

Summary

Summary

Quality Assurance in Debian

Debian :

- the largest volunteer-based GNU/Linux distribution
- renowned for its quality

QA in general plays an crucial role :

- to ensure a minimal quality level for all packages
- to track not-so-well maintained packages
- ...

Quality Assurance in Debian (2)

But some QA tasks require a lot of computing power

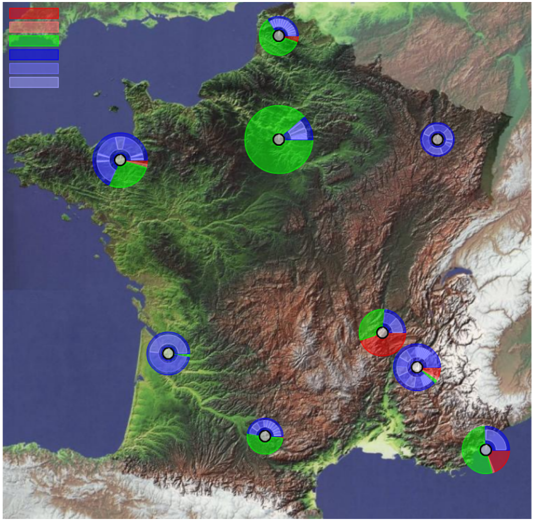
- e.g rebuilding all packages in Debian :
  about 10 days on a single computer

Difficult to perform by volunteers who pay their electricity bills, especially on a regular basis.

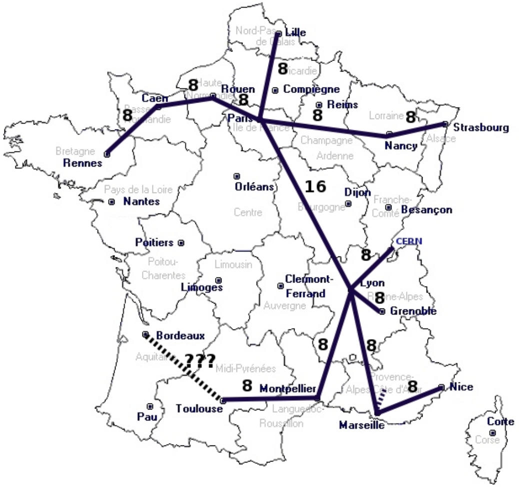
Grid'5000

- aims at building an highly reconfigurable, controlable and monitorable experimental grid
- dedicated to computer science research
- funded by french ministry of research, INRIA, CNRS, ACI Grid, and other public organizations
- gathers 1200 compute nodes (2500 CPUs) in 13 clusters
- typical node : Dual-Opteron 2 Ghz, 2 Gb of RAM
- high speed network (10GbE)
- free time-slots during nights and week-ends

Grid'5000 (2)



Grid'5000 (3)


Lille
Nord-Pas de Calais
8
Picardie
Compiègne
8
Haute Normandie
Rouen
8
Caen
Basse Normandie
8
Paris
Ile de France
Reims
8
Lorraine
8
Strasbourg
Alsace
Nancy
Champagne Ardenne
Bretagne
Rennes
Orléans
16
Dijon
Pays de la Loire
Nantes
Centre
Bourgogne
Franche-Comté
Besançon
CERN
Poitiers
Poitou-Charentes
Limousin
Limoges
Clermont-Ferrand
Auvergne
8
Lyon
Rhône-Alpes
8
Grenoble
Bordeaux
Aquitaine
???
Midi-Pyrénées
8
8
8
Montpellier
Provence-Alpes Côte d'Azur
8
Nice
Pau
Toulouse
Languedoc-Roussillon
Marseille
Corte
Corse

(Obvious) idea : use Grid'5000 to work on Debian QA

- Which tests are suitable ?
- With which infrastructure ?

Summary

QA tasks performed on Grid'5000

Ideal task :

- consumes a lot of time
- can be distributed over a lot of nodes
- doesn't generate too many false positives
- would improve Debian quality

Two different tasks performed on Grid'5000 :

- Rebuild of all packages in Debian
- Installation and removal testing using Piuparts

Rebuilding all packages in Debian

- `Arch :all` packages are only built on the developer's machine
- `Arch :any` packages are only built automatically before they reach unstable

After that, the build environment changes :

- newer/older compiler and libraries
- build-dependencies removed

Not tested automatically, but important for the release :
Etch must be *self-contained* (think of security upgrades !)

Easy to distribute (build in parallel)

Installation and Removal testing

installability can be tested statically (see `debcheck`, `edos-debcheck`) But packages have *maintainer scripts* :

- executed during package installation and removal
- to configure stuff, start services
- helper scripts exist (`debconf`, `update-{rc.d,modules,inetd}`)
- lots of bugs : missing dependencies, shell scripting mistakes, etc

Installation and Removal testing (2)

`piuparts` automatically :

- installs packages in a near-empty chroot
- remove it
- remove as many packages as possible
- purges it

⇒ most extreme test for maintainer scripts

But quite a lot of false positives :

- packages that prompt without debconf
- packages that depend on a DBMS (mysqld,...)

Easy to distribute (test packages in parallel)

Summary

Infrastructure for QA tests on Grid'5000
Principles

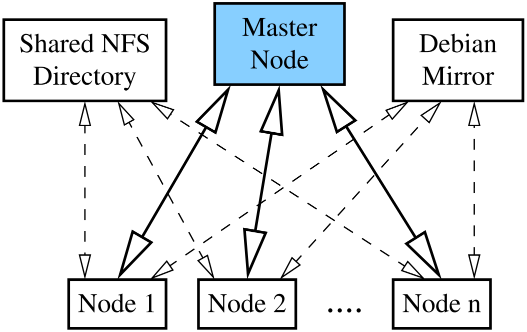
- connection to Grid'5000 nodes via SSH
- one task per node (easier to manage)
- simple master/slave architecture

Infrastructure for QA tests on Grid'5000

Architecture

3 central points :

- *Master* node that schedules jobs
- Shared NFS directory to write results
- Internal Debian mirror



Infrastructure for QA tests on Grid'5000

Typical job (piuparts test)

- 55 nodes are reserved ; deployment of a Debian Sid environment using **Kadeploy** is started.
- After 12 minutes : environment deployed on 43 nodes. First node is used as master node :
  - Prepares the other nodes (install required packages, etc)
  - Locally updates the chroots
  - Script responsible for controlling the other nodes is started
- After 2 minutes, preparation is finished : master nodes starts to schedule jobs on the other nodes.
- After 3 hours and 46 minutes, the 18156 packages in etch have been tested

Summary

Results - Grid'5000 bugs

Those experiments allowed to find a few important problems on Grid'5000 : misconfigurations, performance problems, etc.

In the future, it will serve as a testcase to validate extensions to the platform

Results - Debian Bug Reports

About 200 RC bugs found (and fixed) in Debian Etch

- about 100 from rebuilds
- about 100 from piuparts testing

Efforts welcomed by a majority of developers (but not all :-)

Results - speed-up

Rebuilding the 10217 packages in Debian Etch :
about 10 days on a single computer

⇒ about 7.5 hours on Grid'5000

Testing the 18153 binary packages in etch :
about 5 days on a single computer

⇒ about 3 hours and 46 minutes on Grid'5000

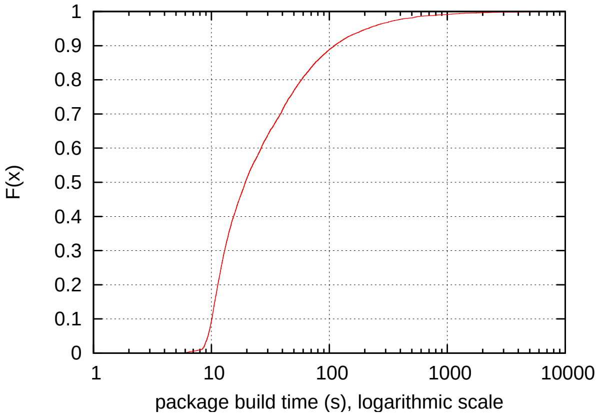
Summary

Future Work

- Improve the infrastructure :
    - Jobs using several Grid'5000 clusters at the same time
    - Central Debian mirror is a bottleneck
      $\Rightarrow$ local cache on the nodes
    - Shared NFS directory for logs is a bottleneck
      $\Rightarrow$ try other solutions
- Other QA tasks (less critical ones)
- Increase the rebuild speed-up

Increasing the rebuild speed-up

Most packages take a very short time to build, but a few packages take a very long time (hours)

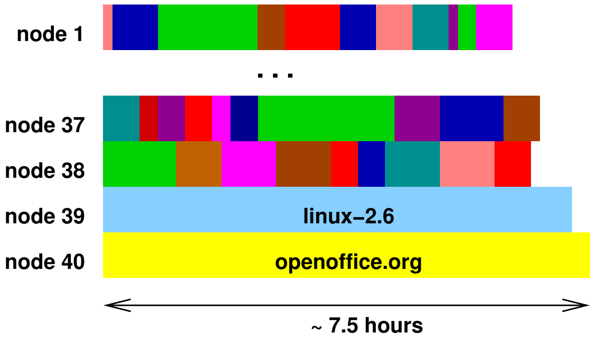


Increasing the rebuild speed-up (2)
Top ten packages

Source package	Time
openoffice.org	7 h 14 min
latex-cjk-chinese-arphic	6 h 18 min
linux-2.6	5 h 43 min
gcc-4.1	2 h 52 min
gcj-4.1	2 h 44 min
gnat-4.1	1 h 52 min
gcc-3.4	1 h 50 min
installation-guide	1 h 45 min
axiom	1 h 44 m
k3d	1 h 39 min

Increasing the rebuild speed-up (3)
Using more nodes is useless

- Already scheduling longest builds first



Increasing the rebuild speed-up (4)
Possible solution : "make -j"

- Grid'5000 nodes have several CPUs, but only one is used during build
- No standard way to tell "*use more than one CPU*" (Debian bug #209008)
- Some packages fail to build when told to use several CPUs

$\Rightarrow$ Possible solution :
only work on the few packages that annoy us...
or just ignore them.

Real bottleneck : manpower for log reviewing

So many logs, so little time...

Such QA tasks were traditionnally *solitaire* games

Sharing the load is necessary to continue on the long term

Summary

Conclusion

Grid'5000 :

- a really nice tool
- well suited to running such tasks

Quality Assurance in Free Software projects :

- could really benefit from using such a tool
- needs improvement, both
  - technically : better testing tools, less false positives
  - also human problem : needs collaboration on reviewing generated data