Debian Quality Assurance on Grid'5000

Lucas Nussbaum

ATER Université Lyon 1
LIP, équipe RESO
(Debian Developer since 2005)

Debian

- A GNU/Linux distribution
  - Like Red Hat, Ubuntu, Fedora, OpenSUSE
- One of the largest volunteer-based organizations
  - 1000+ developers, many more contributors
- One of the largest collection of free software
  - 12000+ source packages, 24000 binary packages
- Many derivative distributions (e.g Ubuntu)

$\Rightarrow$ **An important role in the Free Software world, and many interesting scalability issues**

Debian Quality Assurance

Goal :
Ensure that all packages meet a given quality standard

12000+ source packages !
$\Rightarrow$ even the simpler tests will take a long time
and developers are volunteers !

$\Rightarrow$ **Use Grid'5000 to find bugs in Debian**

2 tests :

- Can all packages be installed, upgraded, removed ?
- Can all packages be rebuilt from source ?

Can all packages be installed and removed ?

Each package depends on other packages
Q : Can all dependencies be satisfied ?
$\Rightarrow$ Can be determined statically (PPS, Univ. Paris 7)

But installation also involves some scripts (bugs ?)
Only way to find bugs : install and remove packages

Piuparts : Debian tool to automatically install, upgrade and remove packages in a clean *chroot*

- Several Piuparts runs on Grid'5000 before the release of Debian 4.0 'lenny'
- About 200 bugs filed and fixed

Can all packages be rebuilt from source ?

Rebuilding packages from source :

- Mandatory before releases
  (security updates, legal issues)
- Allow to detect many problems
  - Compatibility issues
  - API changes
- Stress-test the packages used to build (toolchain)

Interesting test for Grid'5000 :

- Can be fully automated
- CPU- and IO-intensive

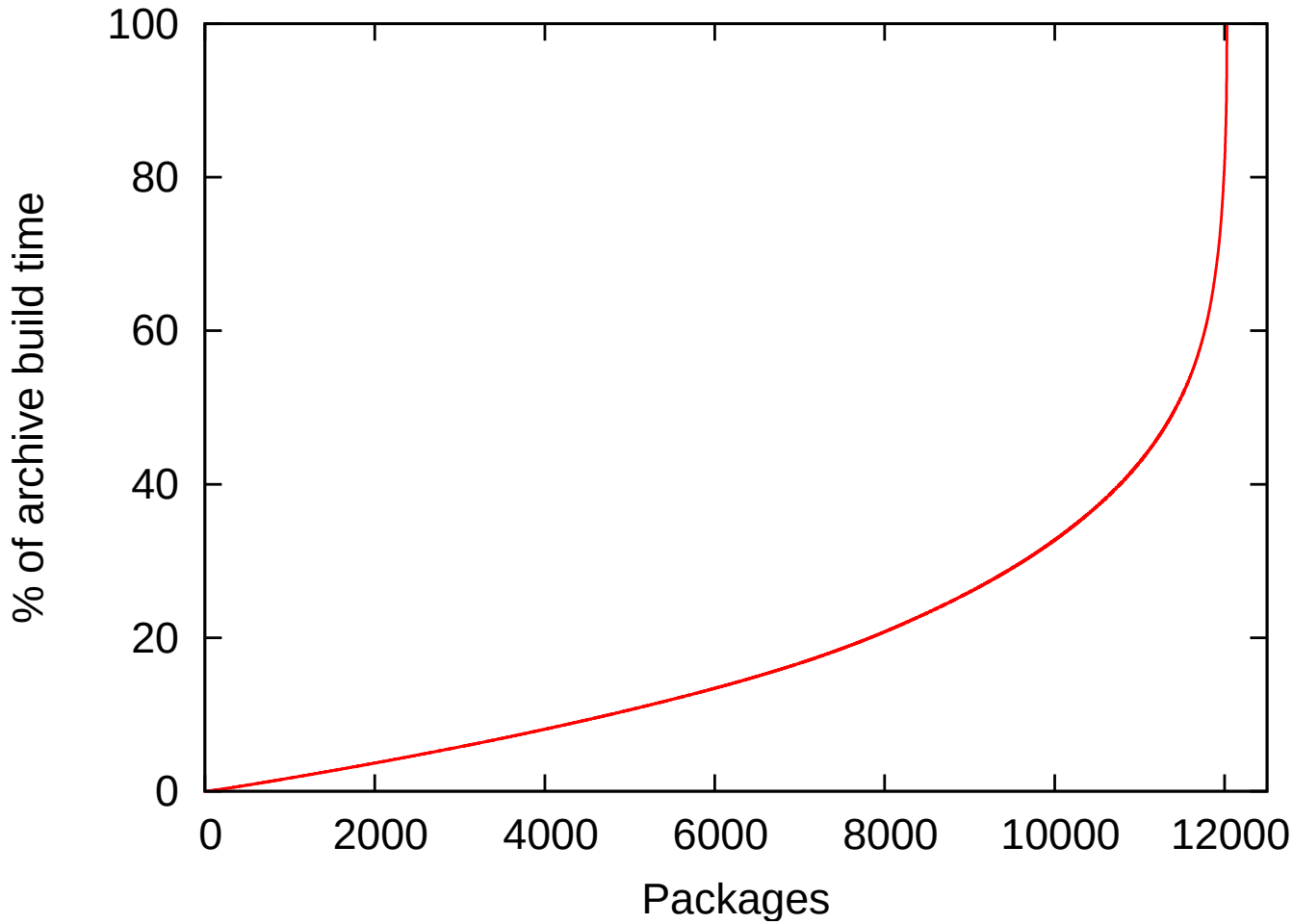
Rebuilding Debian on Grid'5000

On a single (modern) computer : 2 weeks

Difficult to port efficiently to Grid'5000 :

- Complex infrastructure required
  - Debian mirror, *chroot*, root access
    Specific tools : `sbuild`, `schroot`
- Not trivial to parallelize
  - Very different build durations
- Needs to be reliable

Packages build time



5% of the packages take 50% of the build time

Longest builds

Package	Time
openoffice.org	7 h 33 m
openjdk-6	5 h 42 m
insighttoolkit	5 h 38 m
gecode	4 h 51 m
latex-cjk-chinese-arphic	4 h 38 m
linux-2.6	4 h 33 m
gcc-4.3	4 h 21 m
gcc-4.2	3 h 38 m
installation-guide	3 h 28 m
qt4-x11	2 h 12 m

Rebuild infrastructure

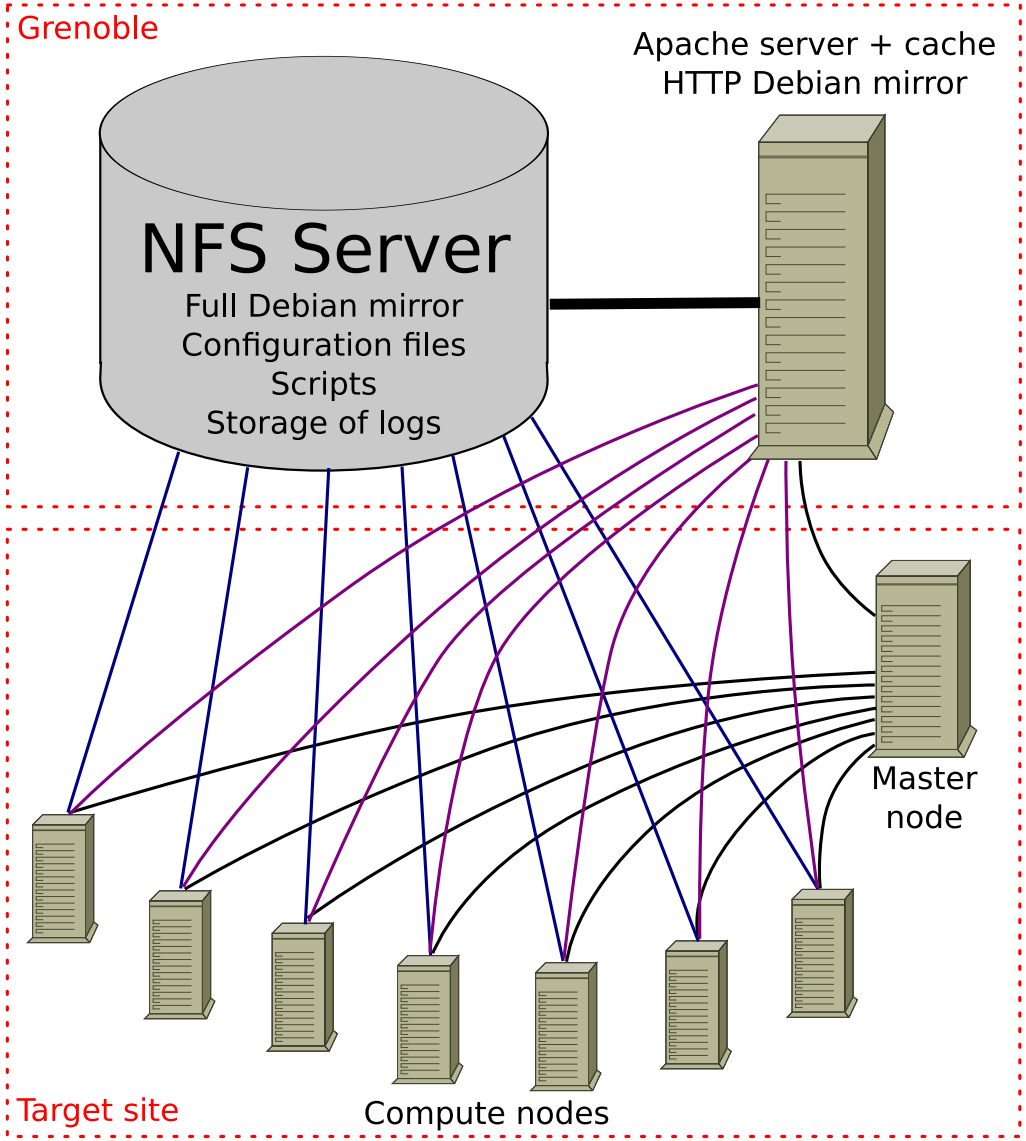
2 parts :

Static part (Grenoble)

- NFS server
- HTTP mirror (VM)

Dynamic part (target site)

- Master node (schedules the tasks)
- Build nodes



Setup steps

1. Nodes are reserved using OAR
2. Nodes are deployed using Kadeploy (+Katapult)
   - Standard environment used (sid-x64-base-1.1)
3. A script is copied from the frontend to one node, then executed. NFS directory is mounted, then another script is executed from there.
4. A script located on the NFS directory is executed to start the script that controls the master node
5. From the master node, a script is copied to all nodes, then executed to prepare the other nodes
6. From the master node, tasks are started using SSH

Katapult

Simple, well-tested script to :

- Deploy nodes
- Check that they work properly
- Re-deploy nodes if not enough nodes have been deployed
- Copy SSH key to the nodes
- Run a script with successful nodes

Example usage :

```
oarsub -t deploy './katapult -e my_env
-copy-ssh-key -sleep - my_script'
```

Google -> Katapult Grid'5000 (2nd result)

Use of a standard environment

Use of sid-x64-base-1.1 instead of a customized environment :

(+) Available (and tested) everywhere

(+) Possible to Customize in a script

(-) Outdated most of the time

(-) Debian sid is a moving target, might break your scripts

(-) Customizing in a script is much harder than customizing interactively once

Use of NFS to share files

- Not scalable
- Poor performance with high latency
- ... but so easy !

$\Rightarrow$ Useful for parts of experiments where performance is not critical

Results

- Full rebuild of Debian in less than 8 hours (about 60 nodes, blame OpenOffice)
- 1000+ Debian bugs filed and fixed
- Used to test future changes in Debian ("*What would it break if we... ?*")
- Used to test future GCC and binutils releases
- Also helped to find many Grid'5000 bugs ;)

Future work

Optimizations :

- Running several builds per node
  - Must not increase the length of long builds !
- Disk I/Os
  - We could do everything in memory
  - But applications are not helpful
  - tmpfs ? ext4 ?

Debian Quality Assurance on Grid'5000

Lucas Nussbaum

ATER Université Lyon 1
LIP, équipe RESO
(Debian Developer since 2005)