Rebuilding Debian using Distributed Computing

Lucas Nussbaum

Laboratoire de l'Informatique et du Parallélisme (LIP)
Université Lyon 1 - ENS Lyon - INRIA

(Debian developer since 2005)

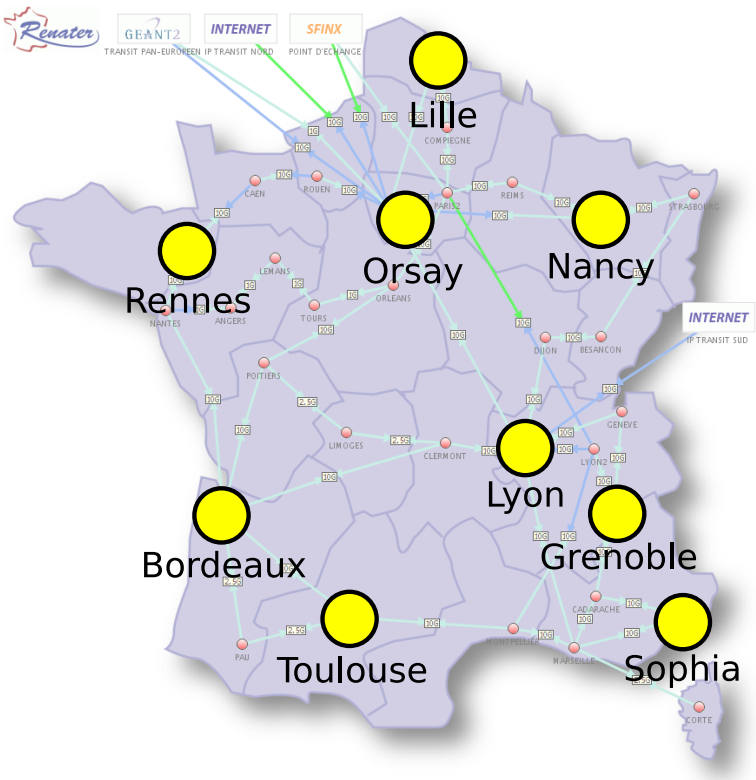
Goal and outline

Present how we were able to execute a complex application on a powerful platform

Outline :

- Execution platform : Grid'5000
- Debian and its Quality Assurance
- Description of the tasks that were executed
- Infrastructure
- Optimizations

Execution platform : Grid'5000



Experimental platform dedicated to research on distributed systems (no production jobs)

9 sites in France, 15+ clusters

1600 nodes, 5000 cores

dedicated network infrastructure
10 Gbps interconnection network

reconfigurable nodes :
deployment of user environment (full system, not virtual machines) using *KaDeploy*
$\Rightarrow$ root access on the nodes

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 !

Main tests :

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

$\Rightarrow$ **How can we use Grid computing to run those tests ?**

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 lab, 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*

- Simple problem (massively parallel)
- 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
  - Bugs introduced by developers
  - Compatibility issues, like API changes
- Stress-test the packages used to build (toolchain)

Interesting test :

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

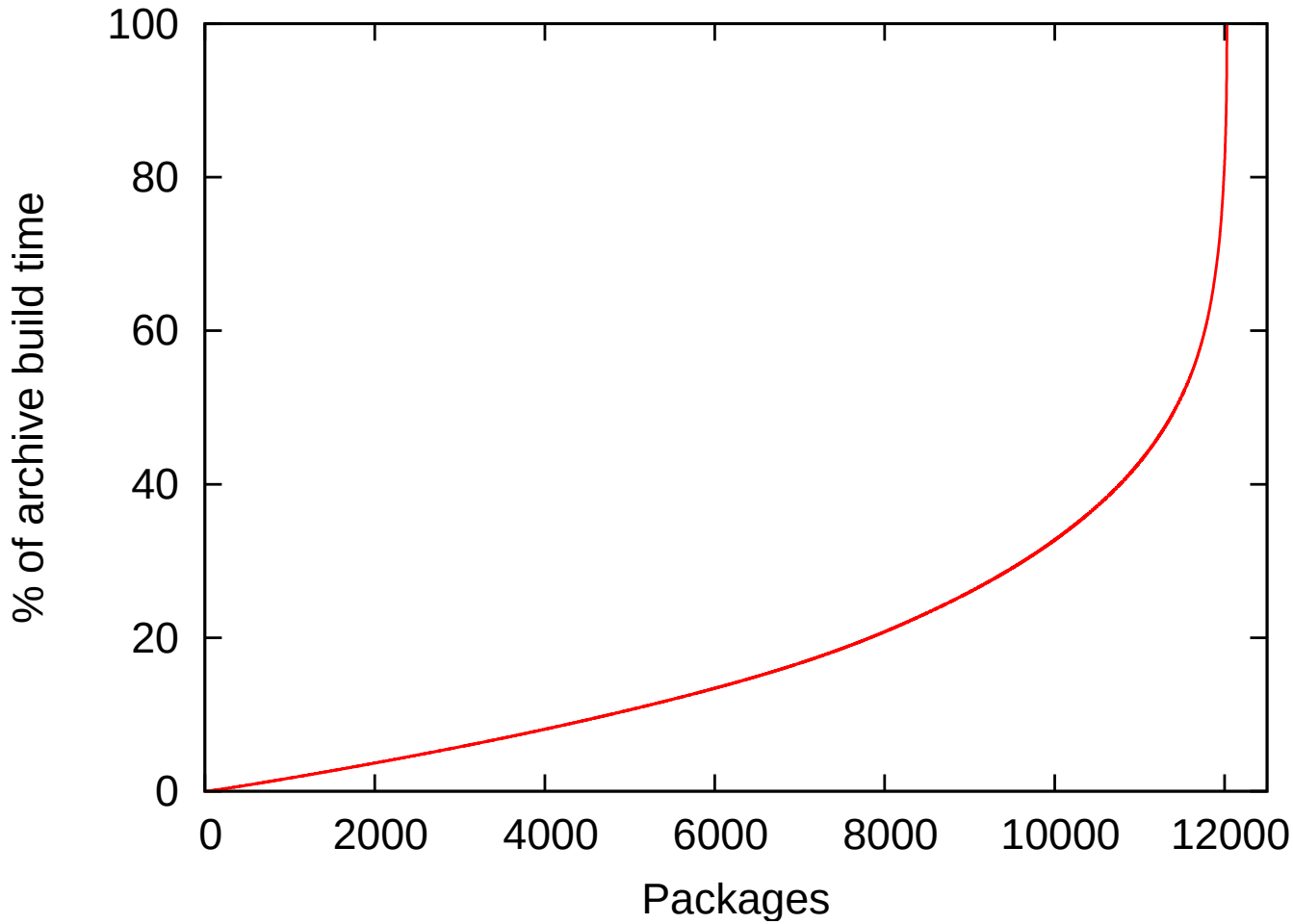
Rebuilding Debian on Grid'5000

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

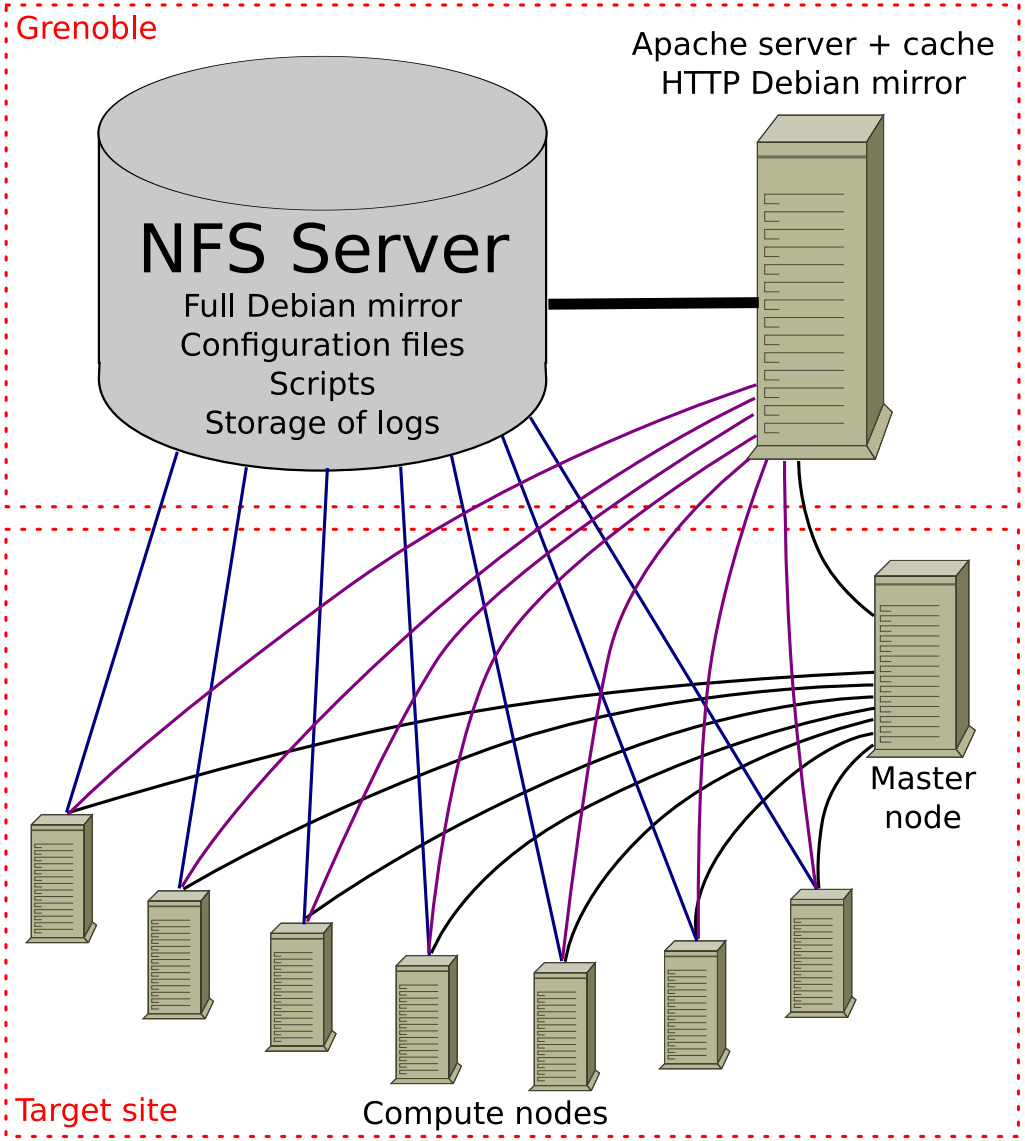
2 parts :

Static part (Grenoble)

- NFS server
- HTTP mirror (VM)

Dynamic part (target site)

- Master node (schedules the tasks)
- Build nodes



Setup steps

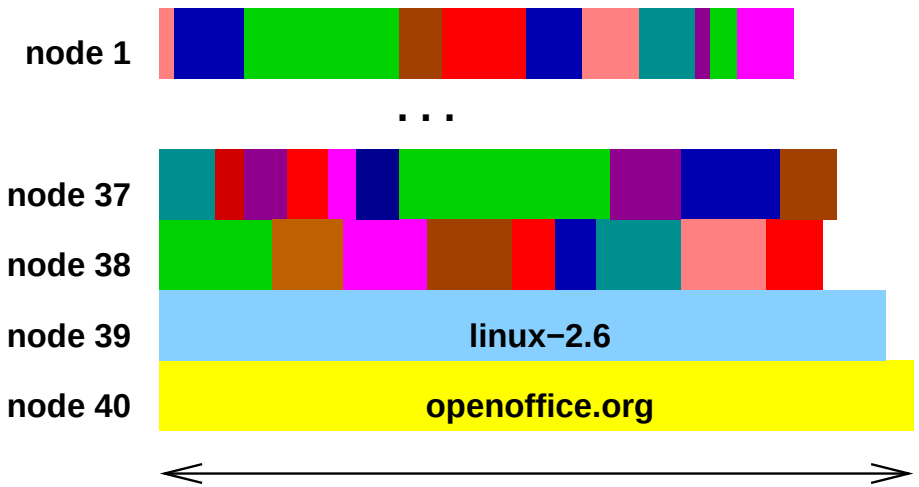
1. Nodes are reserved using the OAR batch scheduler
2. Nodes are deployed with a specific user environment using Kadeploy (+Katapult)
3. Final configuration is performed by a script
4. The master node is started
5. The master node finishes the preparation of the slave nodes
6. From the master node, tasks are started using SSH

Optimizations

Two goals :

- Reduce the walltime
  If useful, we could use more nodes
  - Requires making the longer builds faster
- Increase the efficiency
  use less nodes without increasing the walltime

Scheduling



With enough nodes, walltime = duration of longest build

(Obvious) optimization : schedule longest builds first

Using several cores when building

= "make -j"

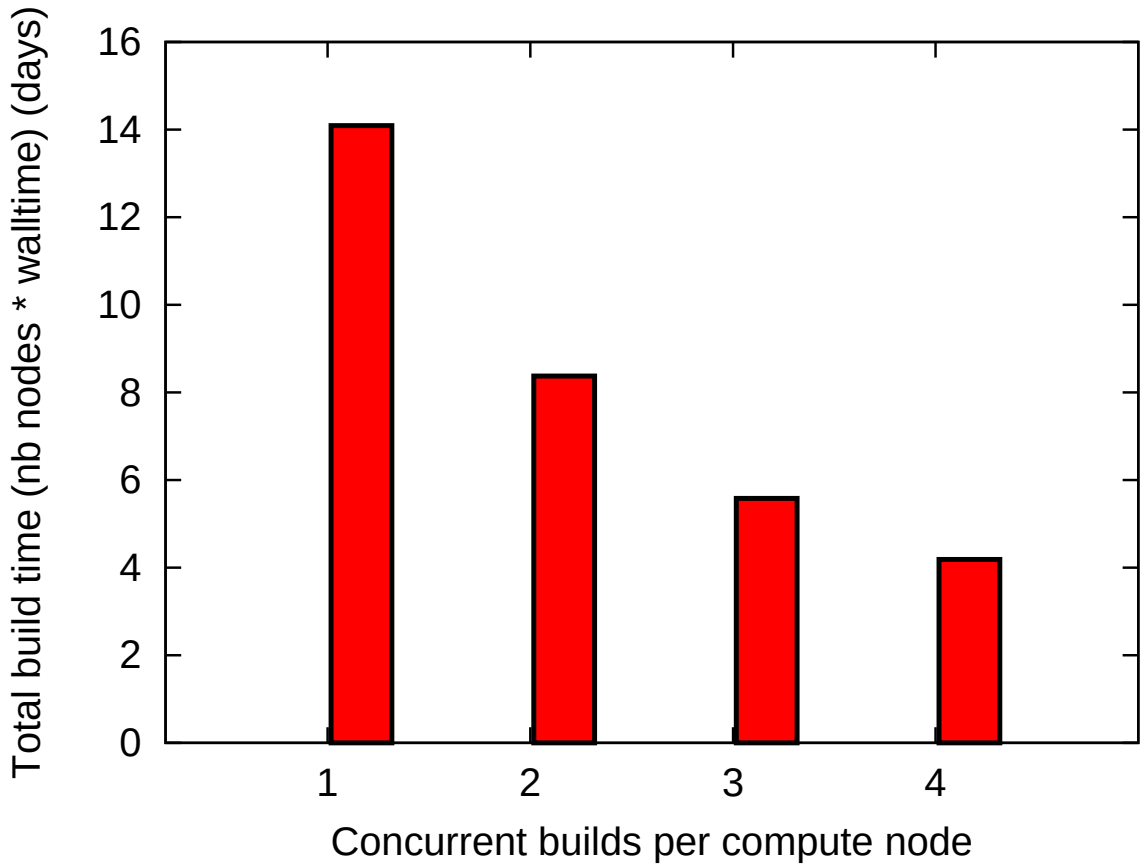
- Not available in most Debian packages
- Difficult to add : unsupported by many build systems
- Implemented in some large packages (OO.org, etc)

Building several packages on the same node

- Parallelism at the global level, not at the package level
- Easy way to make use of several cores per node
- Allows to reduce the number of nodes
- But must not increase the build time of the longer packages
  - Or the walltime would be affected

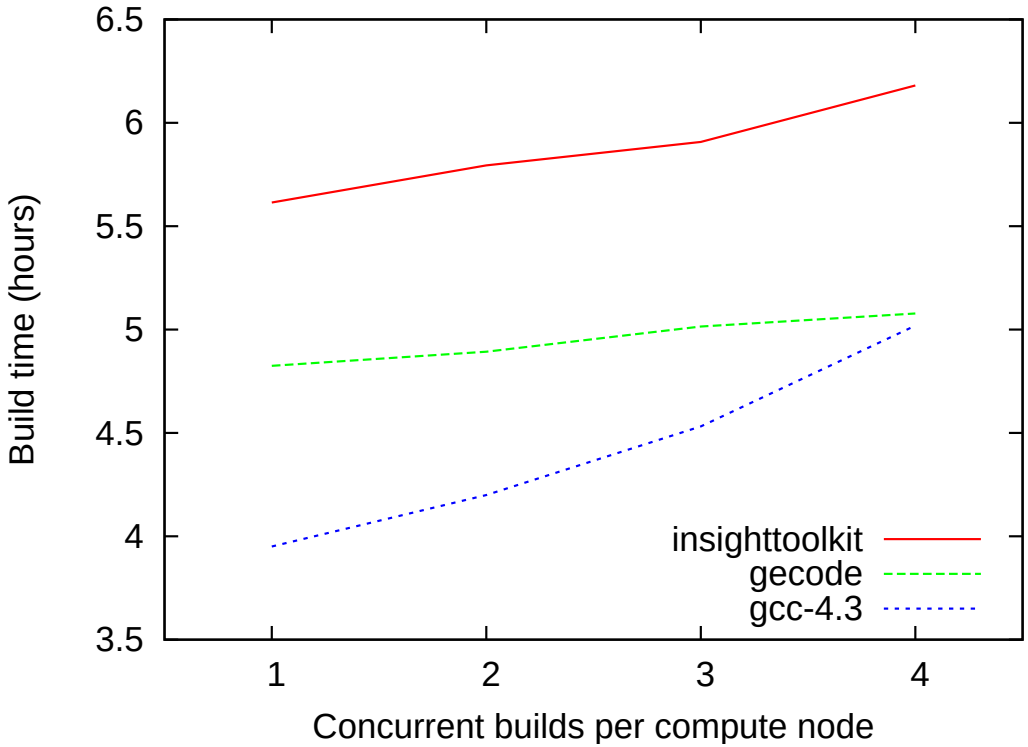
Not as easy as it sounds : I/O and memory bottlenecks !

Total build time



⇒ Building several packages on the same node obviously helps reducing the total build time

Individual packages



But slows down the build of individual packages
Could increase the wall time
$\Rightarrow$ Needs better isolation / prioritization

Improving the I/O bottleneck

- Prefetch reads (read-ahead), **make writes non-blocking**
  - Keep things in memory (as much as possible)
- Classic application : easy to control disk writes (`fsync()`)
- Debian packages building :
  large variety of tools being used :
  compilers, text processors, test suites, ...
  - Impossible to modify all those tools
- Idea : use tmpfs (file system in RAM + swap)

Improving the I/O bottleneck : tmpfs

Using tmpfs :

- Reduces the build time significantly
  - Short builds benefit more than long builds
- But also exposes some bugs
  - Problems with mixing file systems with different time accuracy (second vs nanosecond)

Conclusion

Debian Quality Assurance : complex applications

- Unusual requirements, met by Grid'5000
- Stresses the platform in interesting ways (CPU, I/O)

Provides interesting problems :

Scheduling, Parallelization, I/O optimization

Successful : full rebuild of Debian in less than 8 hours

(about 60 nodes, blame OpenOffice)

1000+ Debian bugs filed and fixed

Impact on the Free Software community

- Used to test possible changes in Debian
- Used to test future GCC and binutils releases

Also helped to find many Grid'5000 bugs

Future work

- Split the build into seperate jobs
  - But cannot deploy the environment before each job
    5 mins to deploy environment / some packages build in 10s
- Only built `i386` and `amd64` packages

  $\Rightarrow$ Build for other Debian architectures using emulators
  e.g use Qemu to build packages on ARM
- Rebuild using already rebuilt packages
  - Instead of using packages already in the Debian archive
  - Requires a lot more work on scheduling
  - Not always possible

Rebuilding Debian using Distributed Computing

Lucas Nussbaum

Laboratoire de l'Informatique et du Parallélisme (LIP)
Université Lyon 1 - ENS Lyon - INRIA

(Debian developer since 2005)