The Ultimate Debian Database
Consolidating Bazaar Metadata for Quality Assurance and Data Mining

Lucas Nussbaum* and Stefano Zacchiroli†

* LORIA, University of Nancy, France
† Université Paris Diderot, PPS, France

debian

FLOSS needs : distribution vs development

Most FLOSS projects require some infrastructure

- Often similar needs
- Led to the development of standard *forges*

FLOSS needs : distribution vs development

Most FLOSS projects require some infrastructure

- Often similar needs
- Led to the development of standard *forges*

FLOSS distributions : more complex requirements

- Archive management, build bots, Quality Assurance tools
- No standard solution :
  - Small number of different distributions
  - Different history, backgrounds, processes

  $\Rightarrow$ No interest in converging to a standard solution

FLOSS needs : distribution vs development

Most FLOSS projects require some infrastructure

- Often similar needs
- Led to the development of standard *forges*

FLOSS distributions : more complex requirements

- Archive management, build bots, Quality Assurance tools
- No standard solution :
  - Small number of different distributions
  - Different history, backgrounds, processes

  $\Rightarrow$ No interest in converging to a standard solution

However : Need to combine data inside and between distributions

- Research : focus on data-mining rather than on implementations
- Collaboration between distributions
- Quality Assurance

Data-mining for Quality Assurance

Debian : most important community-driven distribution
27'000 software packages

- Need to identify packages of sub-standard quality

Which critical bugs affect source packages in the testing branch, sorted by decreasing package popularity ?

Data-mining for Quality Assurance

Debian : most important community-driven distribution
27'000 software packages

- Need to identify packages of sub-standard quality

Which critical bugs affect source packages in the testing branch, sorted by decreasing package popularity ?

- Need to combine data about packages, bugs, packages popularity
- Requires a 200+ lines script $\leftarrow$ **really hard**

Data-mining for Quality Assurance

Debian : most important community-driven distribution
27'000 software packages

- Need to identify packages of sub-standard quality

Which critical bugs affect source packages in the testing branch, sorted by decreasing package popularity ?

- Need to combine data about packages, bugs, packages popularity
- Requires a 200+ lines script $\leftarrow$ **really hard**

this presentation :
Ultimate Debian Database
The countermeasure developed inside the Debian project
to enable data mining and Quality assurance

Agenda

1. The Debian Data Hell
   - Why is it so hard to combine data in Debian ?
   - Why are we in that situation ?
2. Ultimate Debian Database : Architecture and Current Status
3. Examples
4. Conclusions

The Debian Data Hell

Several factors contribute to this situation :

- Heterogeneity
- Community inertia
- Services organization

Heterogeneity

Debian infrastructure composed of several different services

- Developped over 16 years, by different people
- With different design choices and technologies

Purpose	Name	Implementation
Archive mgmt	`dak`	Python, PostgreSQL. Text export
build daemons	`wanna-build`	Perl, PGSQL. was BDB
Bug tracker	`debbugs`	Perl, mbox files
devel accounts	Debian LDAP	LDAP, GnuPG keyrings
Devel tracking	`carnivore`	Perl, Berkeley DB.
Packages uploads	—	Mailing list archives.
Upstream monitoring	`DEHS`	PHP, PostgreSQL database
Package popularity	`popcon`	Perl, text dump
Policy conformance	`lintian`	Perl, text dump
Packages tagging	`debtags`	C++, text DB
Package dashboard	`packages.qa`	Python, XSLT
Web site	www.debian.org	WML / static HTML pages.

Community Inertia

Why not rewrite some of those services ?

- There is no central authority to take such decisions
- Developers often like their original design
  Cared about solving their own problem, not about the "greater good"
- The developer community doesn't feel there is a problem :
  Only a few teams experience the data hell
  = the teams that need to combine data

Services organization

Debian infrastructure and hosting is provided through sponsoring

$\Rightarrow$ Services are scattered over the planet

But :

- Services are tightly coupled
- Make their data publicly available

$\Rightarrow$ A lot of hidden inter-service dependencies

A lot of resistance to change :
changing a service always breaks other services

Services organization

Debian infrastructure and hosting is provided through sponsoring

⇒ Services are scattered over the planet

But :

- Services are tightly coupled
- Make their data publicly available

⇒ A lot of hidden inter-service dependencies

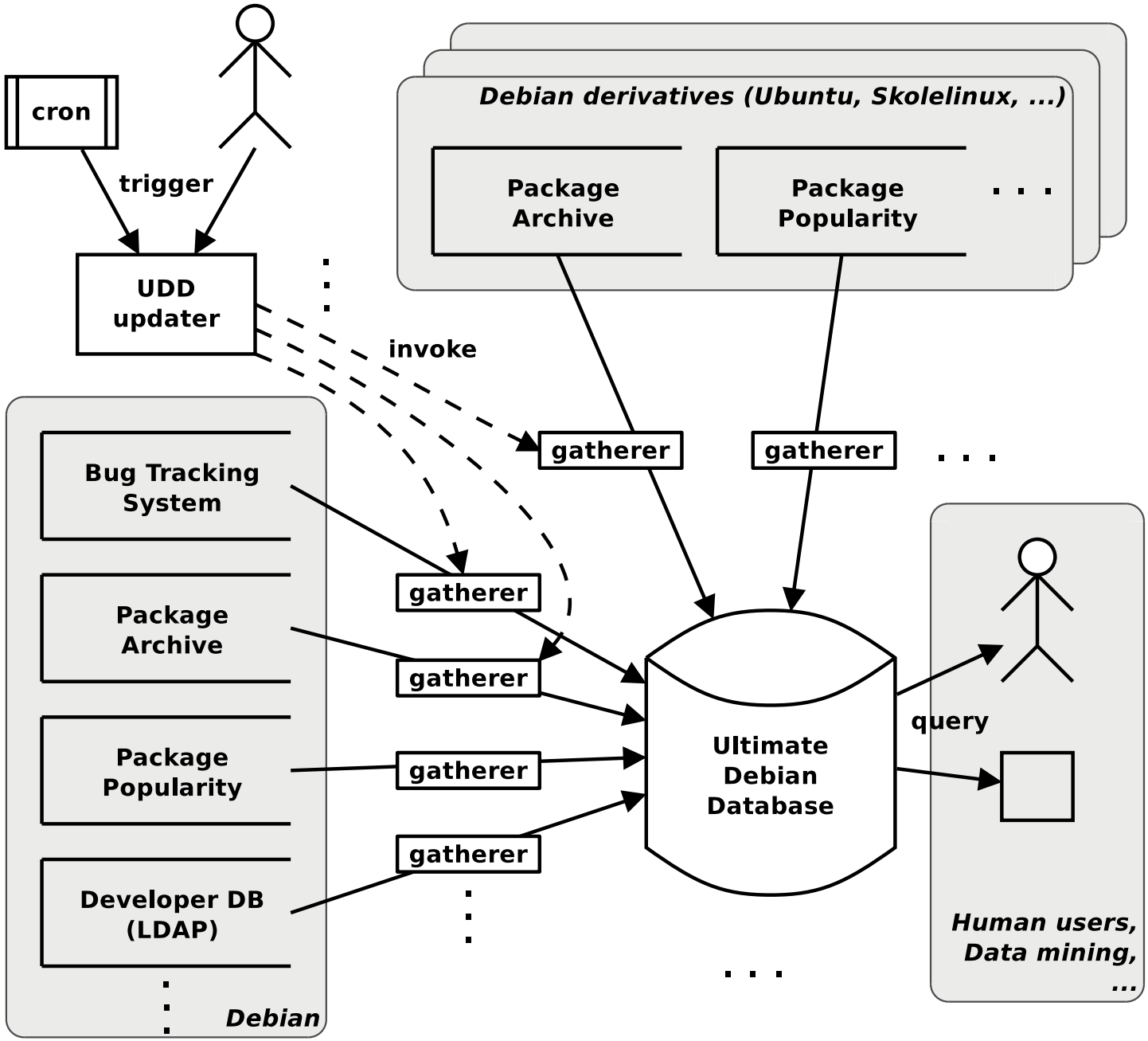
A lot of resistance to change :
changing a service always breaks other services

Ultimate Debian Database

⇒ Tries to Provide a solution to that, based on a single PostgreSQL DB

⇒ But needs to play by the rules of a community-driven distribution
In particular : does not try to replace services, only replicate data

UDD : Architecture



UDD : Design choices

- General-purpose tool, user-friendly
  - No specific applications in mind
    $\rightarrow$ Less optimization opportunities ☹
  - No surrogate keys

UDD : Design choices

- General-purpose tool, user-friendly
  - No specific applications in mind
    $\rightarrow$ Less optimization opportunities ☹
  - No surrogate keys

- Inconsistencies are preserved
  - They might be interesting in some cases
  - When too problematic : hidden using SQL `VIEW`

UDD : Design choices

- General-purpose tool, user-friendly
  - No specific applications in mind
    $\rightarrow$ Less optimization opportunities ☹
  - No surrogate keys

- Inconsistencies are preserved
  - They might be interesting in some cases
  - When too problematic : hidden using SQL `VIEW`

- Full data reloads instead of incremental updates
  $\Rightarrow$ Improves correctness

UDD : Design choices

- General-purpose tool, user-friendly
  - No specific applications in mind
    $\rightarrow$ Less optimization opportunities ☹
  - No surrogate keys
- Inconsistencies are preserved
  - They might be interesting in some cases
  - When too problematic : hidden using SQL `VIEW`
- Full data reloads instead of incremental updates
  $\Rightarrow$ Improves correctness
- (No) storage of historical data
  - Except for some aggregate values
    number of packages, number for each VCS/format
  - But full DB dump is available

Current Status

17 gatherers :

- Source and binary packages for Debian and Ubuntu
- Debian bugs
- Ubuntu bugs
- Packages popularity for Debian and Ubuntu
- Full history of uploads
- Upstream status (DEHS) – is the package up-to-date ?
- Developers identities (*carnivore*)
- Official developers (Debian LDAP)

And packages tags (*debtags*), policy conformance (*lintian*), build daemons status (*wanna-build*), orphaned packages, full history of packages removals, screenshots, localized packages descriptions, new packages under review, status regarding testing migrations, . . .

60 tables, 7 millions of tuples, 3.8 GB

Examples

- Popular yet buggy packages :

```
select sources.source, id, insts, title
from bugs
join sources on sources.source = bugs.source
join sources_popcon
  on sources_popcon.source = bugs.source
where severity >= 'serious'
  and distribution = 'debian'
  and release = 'squeeze'
  and affects_testing
order by insts desc
```

- Tracking of neglected packages and neglecting developers
- Collaboration with Ubuntu
- MSR'2010 challenge

Conclusions

Ultimate Debian Database :

- Attempt at solving the *Debian Data Hell*
- Originally developed for QA and collaboration with derivatives
- Should also ease academic research on Debian
- Makes Debian a better citizen of the FLOSS ecosystem by providing access to most of its data

`http://udd.debian.org/`