Pacman 3

Saul Youssef, Boston University

http://physics.bu.edu/~youssef/

CERN, September 2003

- Pacman is addressing a serious problem in HEP and a critical problem in grid computing.
- Pacman is rapidly being adopted by groups in the U.S., especially grid related projects. The U.S. Atlas testbed, VDT, SDSS, USCMS, LIGO, GLAST, NPACI(Teragrid) and Grid3 are all using Pacman.
- Pacman 3 is about to come out with many major functional improvements. It has the potential to have a significant impact on grid design.

Pacman lets you fetch, install, configure and update software

So it's basically doing

```
./configure; make; make install
```

Or the rpm equivalent, Right? What's the big deal?

Even in the case of a single tarball, complications sneak in…

- You need a way of finding out that you need this particular tarball.
- Which version of the tarball should you use? Where should you get it from? Which tarball is right for your system? Should you use a source or binary tarball?
- What other software do you have to have installed on your system first?
- Installation procedures for tarballs vary. What's the exact procedure for installing the software?
- Do you have to be root to install?
- Once the software is installed are there post installation configuration steps?
- Once the software is installed and configured, how is it to be set up to be used?

So if you figure this out, are you done?

Maybe not…

OOPS! A few Perl modules are missing.

Maybe not…

OOPS! A few Perl modules are missing.

OOPS! You have to upgrade to JDK-1.4.

Maybe not…

OOPS! A few Perl modules are missing.

OOPS! You have to upgrade to JDK-1.4.

OOPS! You have the wrong version of RedHat.

Maybe not…

OOPS! A few Perl modules are missing.

OOPS! You have to upgrade to JDK-1.4.

OOPS! You have the wrong version of RedHat.

OOPS! You have the wrong Linux kernel.

Maybe not…

OOPS! A few Perl modules are missing.

OOPS! You have to upgrade to JDK-1.4.

OOPS! You have the wrong version of RedHat.

OOPS! You have the wrong Linux kernel.

OOPS! Your worker nodes need internet access.

Maybe not…

OOPS! A few Perl modules are missing.

OOPS! You have to upgrade to JDK-1.4.

OOPS! You have the wrong version of RedHat.

OOPS! You have the wrong Linux kernel.

OOPS! Your worker nodes need internet access.

OOPS! The software doesn’t work behind a firewall.

Maybe not…

OOPS! A few Perl modules are missing.

OOPS! You have to upgrade to JDK-1.4.

OOPS! You have the wrong version of RedHat.

OOPS! You have the wrong Linux kernel.

OOPS! Your worker nodes need internet access.

OOPS! The software doesn't work behind a firewall.

OOPS! You have the wrong version of gcclib.

Maybe not…

OOPS! A few Perl modules are missing.

OOPS! You have to upgrade to JDK-1.4.

OOPS! You have the wrong version of RedHat.

OOPS! You have the wrong Linux kernel.

OOPS! Your worker nodes need internet access.

OOPS! The software doesn't work behind your firewall.

OOPS! You have the wrong version of gcclib.

OOPS! You have to upgrade to Python 2.1.

Maybe not…

OOPS! A few Perl modules are missing.

OOPS! You have to upgrade to JDK-1.4.

OOPS! You have the wrong version of RedHat.

OOPS! You have the wrong Linux kernel.

OOPS! Your worker nodes need internet access.

OOPS! The software doesn't work behind a firewall.

OOPS! You have the wrong version of gcclib.

OOPS! You have to upgrade your Python 2.1.

OOPS! The software isn't working because of an http proxy.

Maybe not…

OOPS! A few Perl modules are missing.

OOPS! You have to upgrade to JDK-1.4.

OOPS! You have the wrong version of RedHat.

OOPS! You have the wrong Linux kernel.

OOPS! Your worker nodes need internet access.

OOPS! The software doesn't work behind a firewall.

OOPS! You have the wrong version of gcclib.

OOPS! You have to upgrade your Python 2.1.

OOPS! The software isn't working because of an http proxy.

OOPS! You need X11 on the worker nodes.

Maybe not…

OOPS! A few Perl modules are missing.

OOPS! You have to upgrade to JDK-1.4.

OOPS! You have the wrong version of RedHat.

OOPS! You have the wrong Linux kernel.

OOPS! Your worker nodes need internet access.

OOPS! The software doesn't work behind a firewall.

OOPS! You have the wrong version of gcclib.

OOPS! You have to upgrade your Python 2.1.

OOPS! The software isn't working because of an http proxy.

OOPS! You need X11 on the worker nodes.

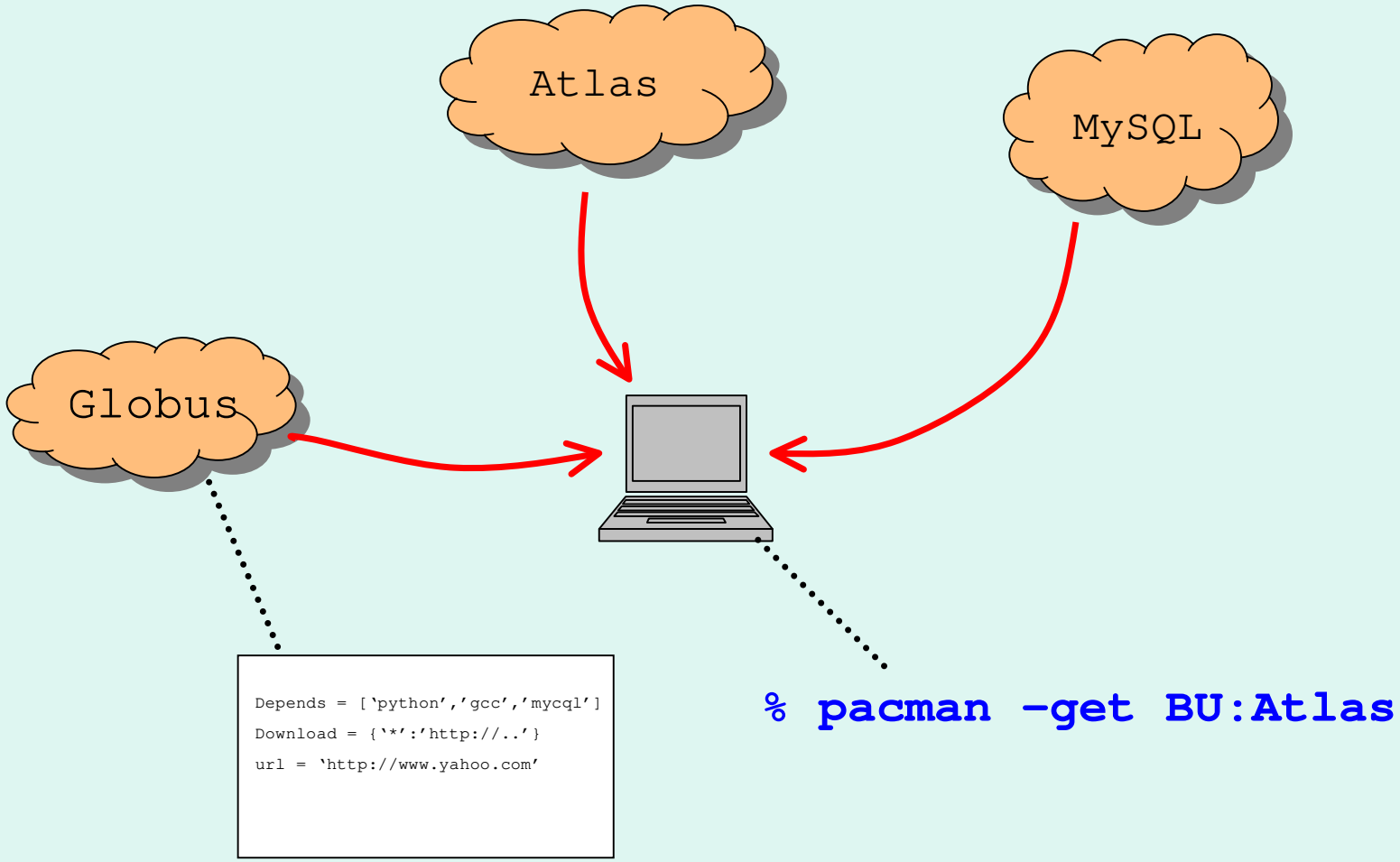
OOPS! You need a different MySQL version installed

Take this, multiply by the number of packages, add the requirement that different sites be consistent and let this vary over time, and you've got yourself a BIG PROBLEM.

A HUGE amount of time is spent in our field going around in circles with these kinds of problems.

This get's worse over time in HEP as we use more and more software. It's an absolutely critical problem for Grid computing.

Pacman helps by changing a basic misconfiguration: the wrong people are working on these problems.



Basic Features of the Design

- Pacman is format neutral: tarballs, rpms, GPT, ups/upd is all OK.
- No cache is used without the installers explicit trust.
- Updates are performed only with the installer's OK.
- Anyone can create caches and this must be easy.
- Language must be simple enough to use very casually but expressive enough for complex installations like VDT or Grid3.

Pacman is used by many and it is spreading rapidly

- U.S. Atlas testbed
- VDT
- WorldGrid
- U.S. CMS for DPE
- SDSS
- LIGO
- GLAST
- Grid3
- NPACI (Teragrid)
- The Globus team is also interested
- CMT broadcast ➡ Pacman, by Simon George and Christian Arnault

There have been many thousands of downloads of pacman.

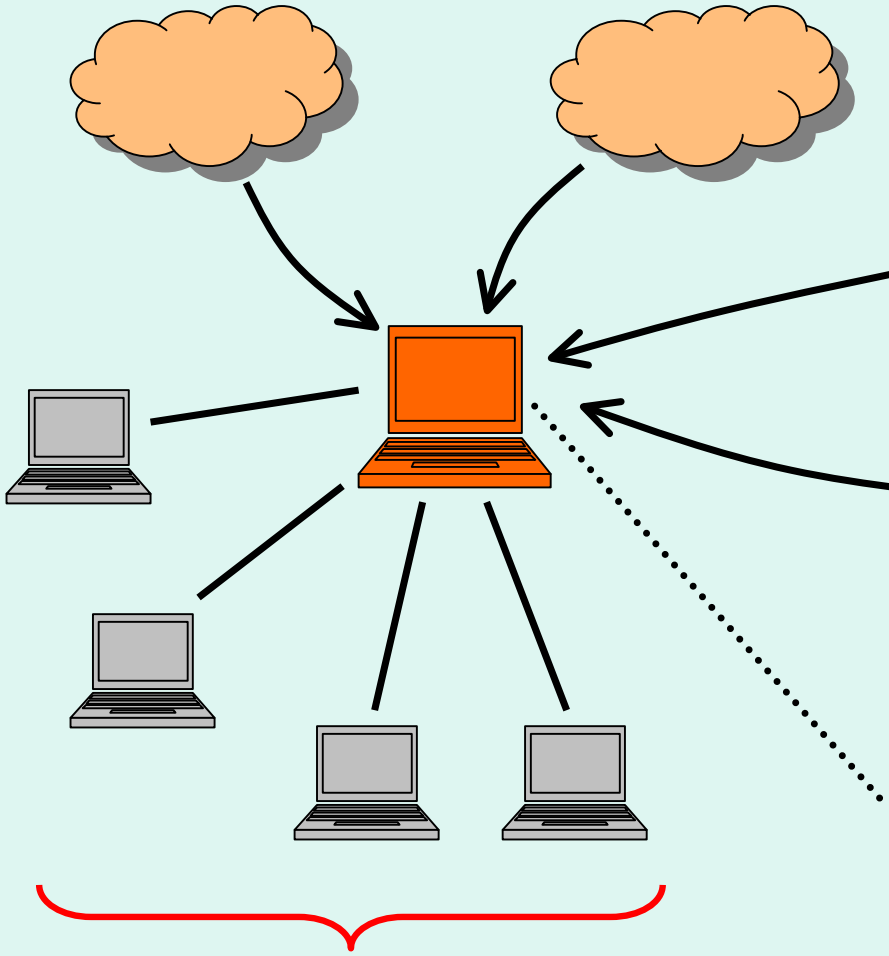
Pacman 2.x is battle hardened. Used in the U.S. Atlas testbed throughout DC1.

Pacman Culture

1. Responsibility for installation, configuration, setup and updating working rests with distributed experts: “cache managers”.
2. Don’t fix a problem locally and tell your friends, let a cache manager fix it since they can fix problems globally.
3. Installations should be untouched by human hands.
4. Web installation instructions, post-installation steps are better pacmanized.
5. When writing a .pacman file for a package, express all dependencies. Each package should install completely on it’s own.
6. Make installations non-root-requiring whenever possible.
7. Do not install in hard-wired locations, e.g. /opt/vdt/.
8. Use native installation and relative installation to gain flexibility in where software can be installed.

PACMAN 3.x is currently being tested…

```
package('BU:Atlas | versionLE(1.0)')
remotes(
cd('?Atlas installation')
cu('nonroot')
```



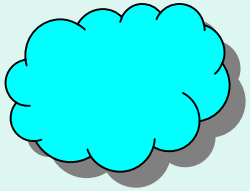
A new (but upward compatible) language.

New kinds of caches

```
% pacman -preview
% pacman -verify
% pacman -repair
```

Greater control and security for the installers

Remote installations, multi-user installations, multi-location installations



New kinds of caches

- **SSH Caches**

  Let's you mix in commercial software. Let's you keep data and/or software private.

- **Snapshots**

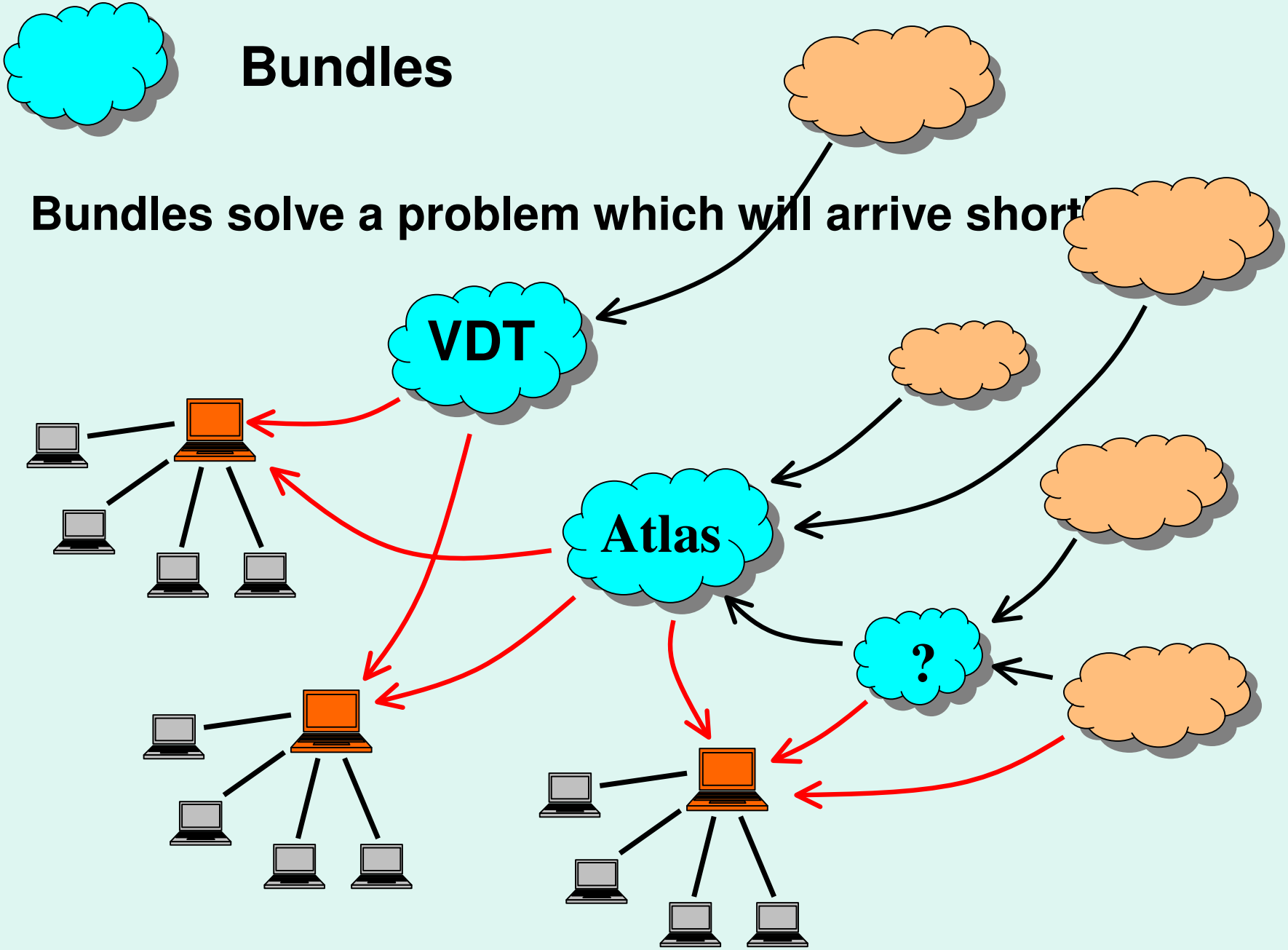
  Very robust archiving

  ```
  % pacman –snapshot BU:AtlasGrid BU:Atlas
  ```

- **Bundles**

  Like snapshots except you can update them and add packages over time.

Bundles



You can depend on packages conditionally

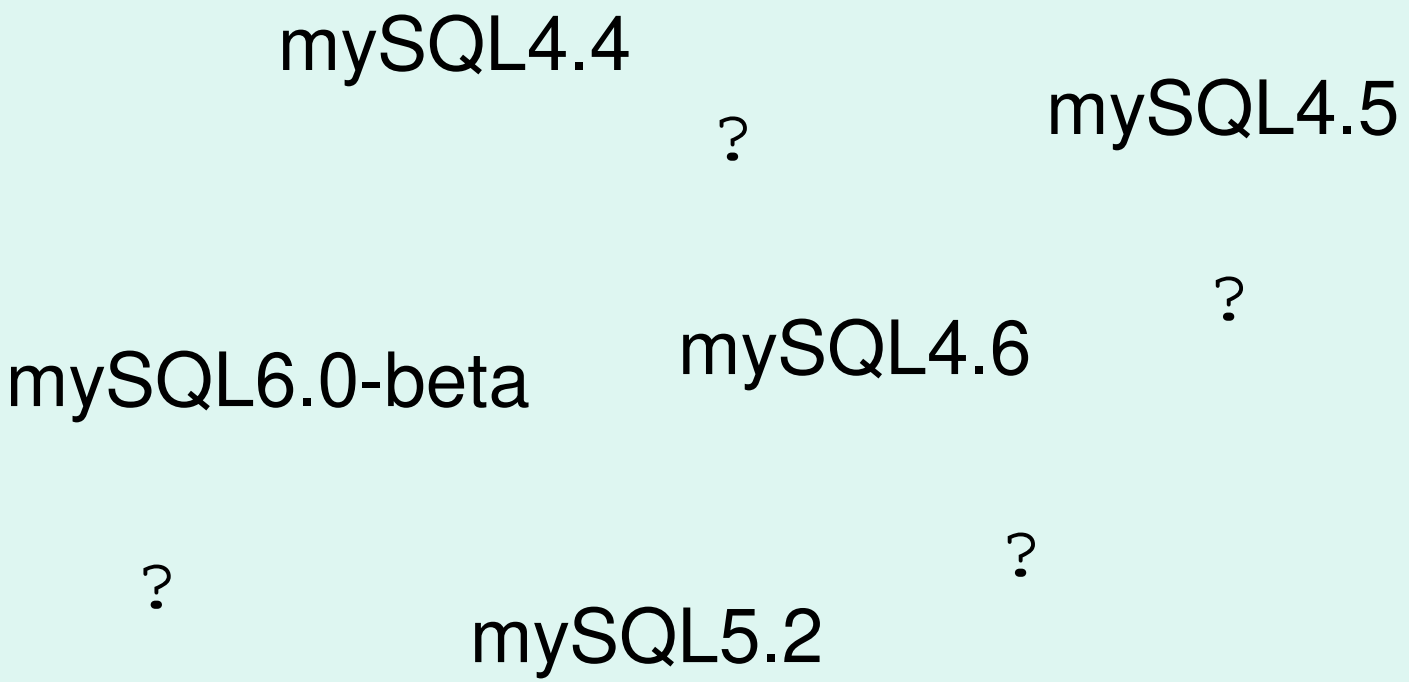
```
packages(`Pacman:Pacman | versionGE(3.0)',
   `MySQL | {versionGE(2.0); versionLT(3.1); OR
             version(4.3.2)}')

% pacman -get "Globus | env('GLOBUS_LOCATION'); inpath('gcc')"
```

Any expression in the same language as the .pacman files are written in.

"Version numbers" are a special case.

What if many packages depend on different MySQL versions?



Pacman 3 searches the trusted caches for a version which satisfies all the requirements.