New Solver Architecture for PkgNG

Vsevolod Stakhov
vsevolod@FreeBSD.org

FreeBSD Cambridge Developer Summit
Computer Laboratory
Cambridge, UK
August 26 – 28, 2013

The problems of the current solver

- Plain dependencies architecture;
- Absence of conflicts resolving/handling;
- No alternatives support;
- Can perform merely a single task: install, upgrade, remove or autoremove, so install task cannot remove packages for examle.

The existing work

There are various of solvers presented:

- `libsolv` - the complete solver library that implements SAT solver based on miniSAT and contains repository handling primitives;
- `apt` solvers interface - an interface to solvers for apt package management system;
- `mancoosi` - a European research project that compares and study different solvers with different optimizations modes, it uses CUDF format for input and output data;

External solvers

To interact with an external solver we have chosen CUDF format:

```
package: devel/libblah
version: 1.0
depends: x11/libfoo

package: security/blah
version: 11
depends: devel/libblah
conflicts: security/blah-devel
```

However, we need to be able not only to call for an external solver but to solve package management tasks internally which cannot be done with the current solver.

How to implement a solver

Many of the contemporary solvers that are used by other package management software, such as yast or zypper, are based on Boolean Satisfiability Problem. It is a well researched problem (NP complete) that searches for a solution for the given Boolean formula. Boolean problem should be normalized for all algorithms and is a conjunction of clauses, each of clause is a disjunction of variables itself, for example:

$(x_1 \| \neg x_2 \| x_3) \& (x_3 \| \neg x_1) \& (x_2)$

Making a SAT problem

It is possible to convert a request to the packages management system to a SAT problem by converting conflicts and depends to the disjunction clauses. We need to perform the following steps:

- Assign a variable to each package: package A $\rightarrow a_1$, package B $\rightarrow a_2$
- Interpret a request as a set of unary clauses:
  - Install/Upgrade package A $\rightarrow (a_1)$
  - Delete package B $\rightarrow (\neg a_2)$
- Convert dependencies and conflicts to disjuncted clauses

Converting dependencies and conflicts

- If we have a dependency for package A from package B that is provided by packages B_1 and B_2 (which may be different versions of the same package), then we can either have package A not installed or any of B installed: $(\neg A \| B_1 \| B_2)$
- If we have a conflict between versions of B (B_1, B_2 and B_3) then we must claim that merely a single version can be installed: $(\neg B_1 \| \neg B_2) \& (\neg B_1 \| \neg B_3) \& (\neg B_2 \| \neg B_3)$

The solving of SAT problem

The SAT problem itself is NP complete and assume the full depth search. Luckily there are common tricks to reduce the problem complexity by skipping some paths. Moreover, for packages we assume to avoid deinstall packages till they are not in conflict with the requested ones.

- Trivial propagation - solve unary clauses;
- Unit propagation - solve clauses with only a single unsolved variable;
- Conflicts learning - if we assign some free variable and detect a conflict during unit propagation, we can fallback and learn that this variable must be inversed;

Solvers and PkgNG

For PkgNG we need to adopt the current jobs handling structure to support SAT solver and external solvers. The following steps are done:

- A request is splitted to install/upgrade and delete requests which could be passed simultaneously to the solver;
- A conflicts between packages are detected with a repository creation;
- All depends, reverse and conflicts of the requested packages are analyzed and the package universe is created;
- Each package is defined by its origin and the digest of significant fields (version, options and so on);

Solvers and PkgNG

- After the universe and the request are formed it is possible to pass them to an external solver using CUDF exporter or to the internal SAT solver by creating a SAT problem corresponding to this package task.
- After solving our request pkgng will be able to install required and delete conflicting packages. Upgrade procedure can be the same: we remove the old version of package and install a new one.

Perspectives

- Using pkg solver for ports management (need to think how to form universe from the ports).
- New dependencies and conflicts: not just from a plain package but from specific versions or variants (such as $libblah > 1.0 + option_1, +option_2 \| libfoo! = 1.1$).
- Better support of multiple repositories (counting that they have shared conflicts).
- Provides and alternatives (support from the ports is badly required).
- We can test various of experimental external solvers and eventually select the best one.

Project Status

- Currently, the new solver is very near for initial testing status.
- The other big task is to adopt ports to the new pkg features and integrate pkgng more deeply to the ports infrastructure, for example for dependencies resolution and install/upgrade/delete tasks solving.
- Eventually we need to improve dependencies and conflicts from the plain structure to the advanced dependency formulas.

Thank you for your attention!

ask questions