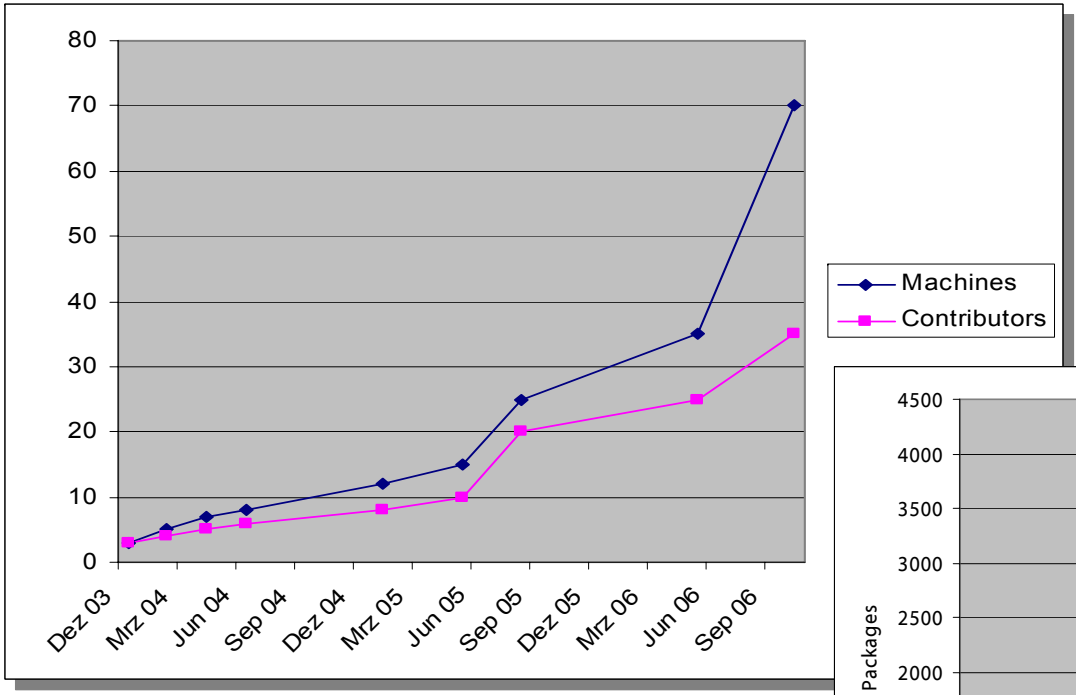
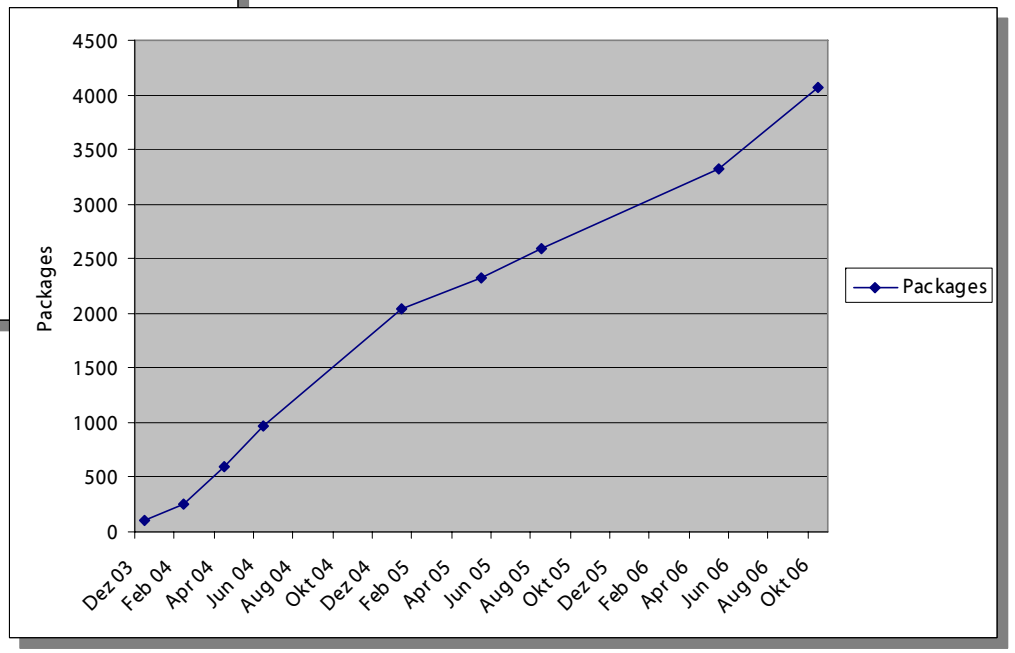
BitBake & OpenEmbedded
past, present, and future



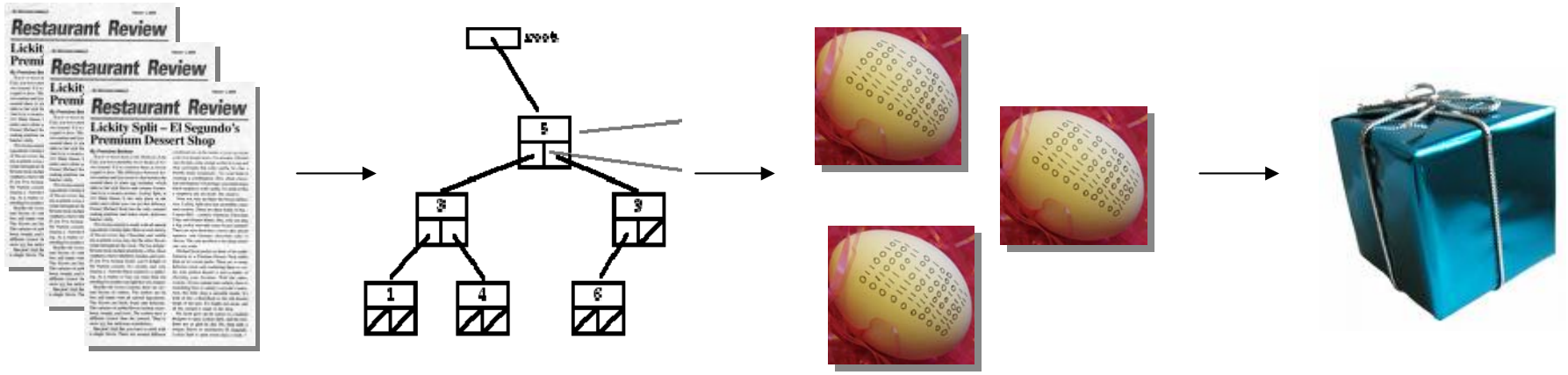


Michael 'Mickey' Lauer
Richard 'RP' Purdie
Holger 'Zecke' Freyther

Agenda

- What‘s it?
- What‘s new?
- What‘s coming?

The BitBake Task Executor




```
# bitbake foo
```

1. parsing data from all recipes it‘s instructed to find
2. For each recipe
   1. Builds a storage area to hold the metadata that comes from the local environment, the recipe itself, the data in the build classes a recipe include.
   2. Computes task dependencies
3. Builds a combined task graph containing all tasks from all recipes
4. Builds all task dependencies for „foo“
   1. generates a shell script on-the-fly out of the metadata
   2. runs the shell script
5. Builds all tasks listed in recipe providing „foo“

openembedded

BitBake Recipe Structure

```
DESCRIPTION = "GNU nano is an enhanced clone of the Pico text editor."
HOMEPAGE = "http://www.nano-editor.org/"
LICENSE = "GPLv2"
SECTION = "console/utils"
DEPENDS = "ncurses"

SRC_URI = "http://www.nano_editor.org/dist/v1.3/nano-${PV}.tar.gz \
           file://glib.m4"

inherit autotools

do_configure_prepend() {
        install -m 0644 ${WORKDIR}/glib.m4 m4/
}
```

- Declarative Language
- Operators: =, =+, +=, ?=, ~=
- Two different kinds of data:
  - Non-executable
    FOO = "bar"
  - Executable
    do_foo() {
        bar }
    FOO = "${@python code here}

- Three different file types:
  .conf -> configuration data
  .bbclass -> (build) class
  .bb -> package recipe
  .inc -> include file
- Common tasks:
  - fetch, unpack, patch
  - configure, compile
  - Install, package

The OpenEmbedded Metadata Repository

A repository containing everything* necessary to build software from scratch

- Build classes containing common tasks, e.g. support for buildsystems
  - autotools
  - qmake
  - python distutils
  - gettext
- Machine configurations for
  - common architectures, developer boards
  - PDAs like Zaurus, iPAQ, HTC
  - WebPads like SIMpad, Nokia 770
  - Network Routers like LinkSys WRT54
  - Network Attached Storage like LinkSys NSLU2
  - GSM Phones like Motorola A780, FIC Neo1973
- Distribution policies like packaging, naming, preferred toolchain, versions, ...
- Recipes (over 4000 nowadays) describing how to build software
  - Descriptions & Licenses
  - Source locations & Patches
  - Dependencies

What's new in BitBake?

- More consistency
- Speed and Memory Improvements
- Modularization of core

Consistent depends/rdepends handling

- Reminder
  - Differenciating build time dependencies and run time dependencies
  - foo.bb: DEPENDS = “bar” ~› foo needs bar to build
  - foo.bb: RDEPENDS = “bar” ~› foo needs bar to run
- Problem
  - Run time dependencies don’t necessarily end up in images, because nothing DEPENDS on them, hence they don’t get built at all
- Workaround
  - DEPENDS being a superset of RDEPENDS
  - Ugly and compromising metadata
- Solution
  - Change BitBake
  - BitBake now knows about RDEPENDS and builds runtime providers as well
- Result
  - Clean metadata
  - Automatic RDEPEND handling

Speeding up BitBake, Part 1

- Memory Usage Improvements
  - True CoW for the data store
  - Don't hold all metadata in memory, only current recipe
- Speeding up uncached parsing
  - Profiling
  - Optimize frequently used loops
  - Mini caches for variable expands, functions, OVERRIDES
- Speeding up cached parsing
  - Old-style cache grew up to ~400MB on disk – I/O bound, nearly slower than reparsing
  - Rethink cache, extract only the needed data = 10MB cache
  - Throw away the rest of parsed data, faster to reparse at recipe build time
  - Now near instant rebuild if cache is unchanged
  - Still: Changing a class or a conf file requires full reparse
- Don‘t generate world dependency tree
  - Previously BitBake always computed a dependency tree not considering what was actually requested
  - It no longer does

Speeding up BitBake, Part 2

- Multithreaded builds to make best use of system resources
- Different tasks use different system resources
  - (fetch, unpack, compile) => (network, I/O+CPU, CPU)
- Problems with previous BitBake core
  - Makes the build path up as it goes along after each recipe completes
  - Recipe based granularity, not task based
  - Complicated by multiple providers, try each in order of preference
- Solution
  - Create new datamodel to hold task dependencies (taskdata.py)
  - Taskdata is compiled to form a runqueue
  - Upon failure, update taskdata and build new runqeue from it
- Result
  - Task based granularity
  - Each task runs in its own thread
  - Controlled by BB_NUMBER_THREADS
- Future Plans
  - Indicate task resource usage
  - Match against available system resources

Separating BitBake Frontend/Backend

- First, there was the bitbake executable
  - Using lib/bb/*
- Then there was OE commander
  - PyQt-based command center
- To ease reparsing woes, I wrote the Shell (bitbake -i)
  - Required some refactoring to reuse lib/bb/*
  - Still a lot of quirks due to a non-consistent state
- Lots of users request a nice UI frontend
  - Even more so, now that we have multithreaded building
- First though, we needed a thorough split of BitBake into
  - Backend (building recipies)
  - Frontend (UI for selecting what to build and showing build progress)
    - Later, something more fancy IDE-style
- Result
  - BitBake goes Client/Server
  - Additional benefit: Remote / Distributed building

What's new in OpenEmbedded?

- More Policies
- Less redundancy
- Quality Assurance

S/TASK-BOOTSTRAP/TASK-BASE/

- task-bootstrap ~› minimum amount packages to get a device „up and running“
- Problems
  - „up and running“ is wish-wash
    - different machines have different capabilities
    - different distributions have different requirements
  - People started (ab)using BOOTSTRAP_EXTRA_(R)DEPENDS to put all kinds of things into resulting images -› duplicated work, confusing for new users
  - But: How to define a task-bootstrap that fits all possible combinations of machine and distribution configurations?
- Solution
  - A fine grained mechanism computing dependencies on-demand
  - A distribution.conf requests the features a distribution wants

    DISTRO_FEATURES = „nfs smbfs ipsec wifi ppp alsa pcmcia usbhost"
  - A machine.conf states the machine capabilities

    MACHINE_FEATURES = "kernel26 apm alsa pcmcia bluetooth"
  - ➲ task-base combines MACHINE_FEATURES w/ DISTRO_FEATURES

Unifying autotools site files

- Reminder
  - Autotools contains makefile generator and runs tests on the build platform to check for capabilities and specifics
  - Since we can't run cross-binaries, we feed prepopulated site files containing test results

```
ac_cv_sizeof_int=${ac_cv_sizeof_int=4}
ac_cv_sizeof_int_p=${ac_cv_sizeof_int_p=4}
ac_cv_sizeof_long=${ac_cv_sizeof_long=4}
ac_cv_sizeof_long_int=${ac_cv_sizeof_long_int=4}
```

- Problem
  - Duplicated work, since every tuple of { architecture, byte endian, libc } needs one
- Solution
  - Introduce common site files and generate the appropriate one on demand
- Result
  - Clean metadata
  - Automatic updates

```
arm-common arm-linux common-glibc endian-little mipsel-linux-uclibc
powerpc-linux-uclibc  x86_64-linux armeb-linux arm-linux-uclibc
common-uclibc  ix86-common powerpc-darwin sh-common
x86_64-linux-uclibc armeb-linux-uclibc common endian-big
mipsel-linux powerpc-linux sparc-linux
```

STAGING_BINDIR CHANGES

- Reminder
  - STAGING_DIR is the area where a recipe installs files which may be needed later
  - STAGING_INCDIR contains header files to include
  - STAGING_LIBDIR contains libraries to link against
  - STAGING_BINDIR contains binaries for the build architecture
- Problem
  - One staging directory, three types of binaries: Host, Target, Cross
- Solution
  - Introduce multiple staging directories: STAGING_BINDIR_NATIVE, STAGING_BINDIR, STAGING_BINDIR_CROSS
- Result
  - STAGING_BINDIR_NATIVE and STAGING_BINDIR_CROSS already in PATH
  - Required vetting references to STAGING_BINDIR
  - Allowed simplification of classes like binconfig
  - Removes one barrier to packaged staging
  - Simplifies QEmu usage

DEVSHELL

- Problem
  - Manually recompiling a package in-place needs recreating the OpenEmbedded environment
- First Shot
  - Devshell recipe – generate script that sets up environment
  - Didn't fit well within the framework
  - Needed fiddling for debugging existing packages
- Second Shot
  - Dedicated DevShell task
  - Appears for every recipe
  - Drops to a shell within that recipe's environment
  - Idea of "interactive" tasks within BitBake
- Usage
  - bitbake -c devshell ‹bbfile›
- Future
  - UI-Integration

Debug Packages

- Problem
  - Debug information usually not present in builds
  - Debug builds are _HUGE_
- Solution
  - Debug information can be shipped seperatly
  - Always build seperate debug packages (foo-dbg)
- Effort necessary
  - Change package.bbclass to spit out the .debug binraries and link to the original (5 lines)
  - Add FILES_${PN}-dbg entry to bitbake.conf (1 line)
  - Change 25% of malformed FILES_ entries to catch up
- Result
  - Get started with debugging earlier
  - Improved metadata

Debian Packages

- Problem
  - Debian packaging desirable
  - Ipkg close, but no cigar
  - Making it possible to replace ipkg with dpkg + apt
- Solution
  - Yank ipkg assumptions spread over parts of OE including variable names
    - IPKG_INSTALL ~> PACKAGE_INSTALL
    - IPKG_EXTRA_ARCHS ~> PACKAGE_EXTRA_ARCHS
- Required restructuration
  - Package classes
  - Image classes
- Result
  - Classes now clean for all kinds of packaging
  - Dpkg/apt has no real understanding of compatitble architectures like ipkg
  - Solved by using separate feed per arch and careful feed configuration

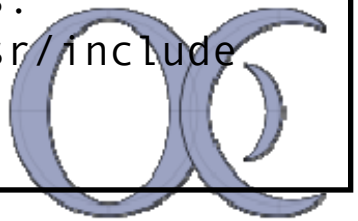
Configuration time QA: Sanity.bbclass

- Problem
  - People hitting same configuration issues again and again
  - OE developers bored of answering the same things
- Solution
  - Write BitBake class that automatically checks the user's configuration
  - Can easily be disabled if you understand BitBake (and not if otherwise ☺)
  - Most core devs thought they'd hate it but have hit useful warnings too
- Result
  - "Please set TARGET_OS directly, or choose a MACHINE or DISTRO that does so"
  - "Please use ASSUME_PROVIDED +=, not ASSUME_PROVIDED = in your local.conf"
  - "DISTRO '%s' not found. Please set a valid DISTRO in your local.conf"
  - "Your installation is missing the following utilitites: GNU make, patch, texi2html"
  - "You do not include OpenEmbedded version of conf/bitbake.conf"

Compile time QA: Fail-Fast Override

- Set of patches to toolchain to make it abort on common mistakes
- Example: Detecting local paths when cross compiling
- Usage:
  - OVERRIDES = "‹...›:fail-fast"

```
| if arm-linux-gcc -march=armv5te -mtune=xscale -DHAVE_CONFIG_H -I. -
I/local/pkg/oe/spitz/tmp/work/armv5te-linux/nano-1.3.9-r0/nano$
-Iintl -DLOCALEDIR=\"/usr/share/locale\" -DSYSCONFDIR=\"/etc\"  -
isystem/local/pkg/oe/spitz/tmp/staging/arm-linux/include  -
I/usr/include -fexpensive-optimizations -fomit-frame-pointer -
frename-registers -O2 -MT$ -MF ".deps/move.Tpo" -c -o move.o move.c;
|       then mv -f ".deps/move.Tpo" ".deps/move.Po"; else rm -f
".deps/move.Tpo"; exit 1; fi
| CROSS COMPILE Badness: /usr/include in INCLUDEPATH: /usr/include
| cc1: internal compiler error: in add_path, at c-incpath.c:362
| Please submit a full bug report,
| with preprocessed source if appropriate.
| See <URL:http://gcc.gnu.org/bugs.html> for instructions.
| CROSS COMPILE Badness: /usr/include in INCLUDEPATH: /usr/include
| Please submit a full bug report,
```



Packaging time QA: Insane.bbclass

- Quality-Insurance
- Inspired by Portage and (early) bug reports
- Post-packaging Checks
  - Package RDEPENDS on a –dbg package
  - .debug directories not in .dbg package
  - ABI and MACHINE of resulting binary don't match
  - Bogus entries in staged pkg-config and libtool files
  - Incorrect permissions of files
  - Security issues with RPATH for binaries
- Usage:
  - INHERIT += insane.bbclass

Using Qemu in/with OpenEmbedded

- Problem
  - Locales had to run on the device, often OOM
- Solution
  - Generate using qemu binary execution
- Qemu can also run systems
- Openedhand developed pseudo machines in Poky
  - MACHINE = "qemuarm"
  - MACHINE = "qemux86"
- Generated images run under Qemu system emulation mode
- Poky has scripts to make this as easy as "runqemu"
- Allow tests to be run on OE images without presence of hardware

Testing Infrastructure (wishlist)

- Regularly compiling different configurations
- Reporting status to central server
- Interconnecting
  - Bugtracker
  - Compile results
  - Installation results
  - Runtime results
  - Commits
- Browser interface to track progress for
  - Single packages
  - Distributions
  - Architectures
- Use monotone to attach test results to bitbake files
- Regression suit
  - Upload package, install it, execute test suite
  - Upload coverage and results
- Do that 24/7

Tiny little things

- New mailing lists @ lists.openembedded.org
  - openembedded-issues
  - openembedded-users
- Monthly Bugsquash Event
  - every last weekend a month
- Removal of MAINTAINERS
- Oe-stylize
- Interactive Patch Resolve Mode
- BitBake & OpenEmbedded goes Debian
  - bitbake.deb
  - task-openembedded-essential.deb

What's coming?

- More Quality
- Support for new OSes
  - OE is supposed to be OS-agnostic
- More package formats
  - rpm
- Stable Snapshots
  - Frequently
  - Stabilization Phase
  - Test Phase
  - Release
- Public buildserver
  - Once remote BitBake access is finished
- OE Book
  - Dual release (german/english)
  - OpenSourcePress.de