ReducingDiskFootprint

This page describes various techniques for developers, derivatives, and OEMs to change an Ubuntu system to reduce the required disk footprint.

Contents

1. Reduce package count
2. Use XFCE instead of GNOME
3. Drop unnecessary packages
4. Drop unnecessary files
    1. Technique
    2. Documentation
    3. Translations
    4. Landscape
5. Compress files
6. Drop login manager
7. Select components of metapackages
8. Select X.org video driver
9. Disable apt caches
10. Trim log files
11. Tune the filesystem
12. Explore alternate filesystems
13. Reconfigure and rebuild the kernel
14. Remove unnecessary kernel modules
15. Other potential possibilities

Reduce package count

The most obvious technique to reduce disk space is to strip out all unnecessary software packages from the install image. For example, a small system typically does not require an on-disk office suite, so eliminating OpenOffice.org will result in space savings.

Rather than starting with the default *ubuntu-desktop* meta-package and stripping it down, it is easier to come from the other direction. Start with the *ubuntu-minimal* meta-package and slowly add packages until the base requirements are met.

Configuring the image to eliminate packages with mere "Recommends" priority is another important technique.

Use XFCE instead of GNOME

The default Ubuntu installation uses the GNOME desktop environment which has quite a large set of dependencies and related disk space cost.

Xubuntu provides the XFCE4 desktop environment, which is still highly functional and provides ability for customization. A default Xubuntu installation weighs in at 1.6G, versus the 2.1G footprint of Ubuntu.

If you use XFCE4, you can drop `libgnome2-0`'s dependency to `gvfs`, to remove `gvfs`, `libgdu`, and `udisks` (about 5 MB). This also helps saving some boot time (order of 0.5 s). The patch was applied to maverick's `libgnome` and can be easily backported to lucid-based systems. On the other hand, if you want a more modern XFCE which stops using the deprecated hal and instead uses udisks and upower, then switch to the xubuntu-dev PPA instead.

Drop unnecessary packages

- In lucid-based systems, `libcairo2` pulls in libdirectfb and some dependencies. This is unnecessary, and can be removed with this patch against cairo. This saves about 3 MB.

- Create a custom `ubuntu-minimal-`*project* seed and metapackage which removes unnecessary packages like `netcat-openbsd`, `tasksel`, and `gnupg`. (~ 12 MB)

- Drop the `gnupg` dependency from `ubuntu-keyring`, `gpgv` is enough. The patch is applied in maverick, and can be easily backported to lucid-based systems. (1 MB)

- Ensure that final images do not keep residues of installer related packages. This needs to be configured in e. g. live-helper (`BINARY_HOOK_DESKTOP_MANIFEST_EXCLUDED_PACKAGES`) or cdimage or ubiquity itself. Packages in question are: casper ubiquity ubiquity-frontend-gtk ubiquity-casper ubiquity-ubuntu-artwork cryptsetup ecryptfs-utils dmraid gparted kpartx reiserfsprogs xfsprogs os-prober parted redboot-tools python-icu libicu42 dmsetup

- If you use a lucid-based XFCE and are not concerned about user data migration, drop the xfconf 4.6 migration script. This removes both a perl script during boot (improving boot speed) as well as 3 MB of perl library dependencies. The patch was applied in maverick and can easily be backported to lucid-based systems.

- If you only need a small XFCE environment, change `pango1.0` to not depend on defoma. This helps getting rid of `perl-modules` and `perl`. (~ 30 MB)

- `synaptic` (a dependency of `update-manager` and others) strictly depends on `scrollkeeper`, which pulls in a lot of XML and Perl stuff. Dropping this dependency helps getting rid of `perl-modules` and `perl`. (~ 40 MB, but of course 30 MB of those are perl and perl-modules, the size of which is already covered in the previous point)

- Printing related packages pull in defoma/perl in lucid. `ghostscript` and `gsfonts` in maverick moved to use `update-gsfontmap` instead, and can be backported to lucid. `cups` unnecessarily depends on Perl, which was fixed in Maverick and is trivial to backport.

Drop unnecessary files

Technique

If it is acceptable for the project to not install local documentation, manpages, include files, etc., these can be ignored.

One-time removal during image build (`livecd-rootfs` in Ubuntu or `live-helper` for OEM) is not enough, since each package update or additonal application installation would bring back the files. So this requires filtering of those files in dpkg.

This is provided by a dpkg patch which recently got committed upstream and got backported into Ubuntu 10.10. Please see `man dpkg` for details about those options. The patch can be backported to Ubuntu 10.04 LTS based projects.

Since the original debootstrap will install quite a lot of base packages without dpkg, and the *project*-`config` package is not necessarily installed before any other package, an additional cleanup during image builds is still required, though. This should remove the same directories/files as the dpkg filter options. These commands can be put into live-helper's `LH_HOOKS` configuration, or into some *project*-`config`'s postinst.

Documentation

- Create a file `/etc/dpkg/dpkg.cfg.d/01_nodoc` which specifies the desired filters. Example:

```
path-exclude /usr/share/doc/*
# we need to keep copyright files for legal reasons
path-include /usr/share/doc/*/copyright
path-exclude /usr/share/man/*
path-exclude /usr/share/groff/*
path-exclude /usr/share/info/*
# lintian stuff is small, but really unnecessary
path-exclude /usr/share/lintian/*
path-exclude /usr/share/linda/*
```

- Remove the same set of files and directories in the *project*`-config`'s postinst. Example:

```
if [ "$1" = "configure" ] && [ -z "$2" ]; then
    echo "Removing documentation..." >&2
    find /usr/share/doc -depth -type f ! -name copyright|xarg
    find /usr/share/doc -empty|xargs rmdir || true
    rm -rf /usr/share/man /usr/share/groff /usr/share/info /u
fi
```

Translations

If you use packages from universe, `/usr/share/locale/` will have a lot of (probably unneeded) translations. If you only need to support a relatively small subset of languages, the unnecessary ones can be filtered out with above dpkg filters:

```
path-exclude /usr/share/locale/*
path-include /usr/share/locale/en*
path-include /usr/share/locale/de*
path-include /usr/share/locale/es*
path-include /usr/share/locale/ja*
path-include /usr/share/locale/fr*
path-include /usr/share/locale/zh*
```

```
find /usr/share/locale -mindepth 1 -maxdepth 1 ! -name 'en' ! -nam
```

If you use XFCE, `/usr/share/xfce4/doc` has translations; these could also be moved into langpacks.

Landscape

If your project is meant to use landscape, you can remove test suites from Landscape dependencies → `/usr/share/pyshared/twisted/test`,

`/usr/share/pyshared/twisted/*/test` in the `python-twisted-{core,web}` packages [3 MB]:

```
path-exclude /usr/share/pyshared/twisted/test*
path-exclude /usr/lib/python*/dist-packages/twisted/test*
path-exclude /usr/share/pyshared/twisted/*/test*
path-exclude /usr/lib/python*/dist-packages/twisted/*/test*
```

Compress files

- The files in `/usr/share/i18n/charmaps` take quite a lot of space (15 MB), and are only required for `localedef`, where some additional CPU overhead does not matter much. This was done in maverick, but the patch can be easily backported to Ubuntu 10.04 based projects.

- apt's indexes in `/var/lib/apt/lists/` are stored uncompressed. Compressing them saves about 26 MB on a system with just the standard binary lucid apt sources, and much more if you also have `deb-src` sources and/or more repositories. There is a branch for supporting compressed indexes which will hopefully land in sid and maverick soon. With this version, you can enable compressed indexes by creating a configuration file `/etc/apt/apt.conf.d/02compress-indexes` (in your *project*`-config` package):

  ```
  Acquire::GzipIndexes "true";
  Acquire::CompressionTypes::Order:: "gz";
  ```

  This currently makes some programs like xapian and synaptics very slow. See the related bug list, so only use this if these bugs are not relevant for you (or better, help fixing them).

- cups' charmap tables in `/usr/share/cups/charmaps` also take some space (3.3 MB). With a relatively small patch they can be compressed and only take 800 kB, saving 2.5 MB. The patch can be easily backported to Ubuntu 10.04 based projects.

Drop login manager

For projects which are only supposed to be single-user with autologin, we do not need to install gdm (which weighs 13 MB together with its extra non-XFCE dependencies). Instead, this could be replaced with a simple upstart script which directly launches a session for the user.

Select components of metapackages

It is common for large upstream meta-packages to depend, in turn, on other meta-packages, in order to create a single conceptual package. Examples are `texlive` which depends on all packages that comprise the TeX life LaTeX distribution, and `openoffice.org` which depends on all individual components (writer, draw, etc.)

If you only need some parts, only depend on the individual components of those.

Select X.org video driver

Ubuntu ships a metapackage `xserver-xorg-video-all` which depends on all available (and supported) video drivers, such as `xserver-xorg-video-apm`, `xserver-xorg-video-ati`, `xserver-xorg-video-vesa`, etc.

In a small-disk-footprint scenario, the hardware platform is typically extremely specific. Thus, it is highly unlikely that the final image will need both the nouveau and the ati video drivers, and it will almost certainly not need the very old drivers like sis, s3, or mga. So you can explicitly seed the needed video driver(s) for the particular -desktop task.

Each video driver provides the virtual package `xserver-xorg-video-`

abiversion, which will satisfy `xserver-xorg`'s dependency.

If your project does not use Tasks (as Ubuntu does), but directly installs the `project-desktop` metapackage, this will still cause `-video-all` to be pulled in. To fix this, you need to modify the `xorg` source package and replace the `-video-all` dependency with something lightweight like `-vesa` or `-fbdev`.

The input drivers are packaged similarly, with the `xserver-xorg-input-all` metapackage. You can explicitly seed a required subset of the drivers and change the `-input-all` dependency to avoid installing all available drivers. Potential candidates for removal are `xserver-xorg-input-synaptics` (for touchpads), `xserver-xorg-input-vmmouse` (mouse in virtual machine guests), and `xserver-xorg-input-wacom` (tablet/touch screen driver).

Disable apt caches

Apt stores two caches in `/var/cache/apt/`: `srcpkgcache.bin` is rather useless these days, and `pkgcache.bin` is only needed for faster lookups with `apt-cache` (software-center has its own cache). Removing those two buys 26 MB, for the price of apt-cache taking an extra two seconds for each lookup.

An image build needs to `rm /var/cache/apt/*.bin`. However, the next `apt-get update` would bring it back, thus the caches need to be disabled in the configuration. Create a file `/etc/apt/apt.conf.d/02nocache` with the following contents:

```
Dir::Cache {
  srcpkgcache "";
  pkgcache "";
}
```

However, please note that if you need to use synaptic, you need to keep pkgcache; see bug 596898.

Trim log files

To avoid piling up too many log files in `/var/log/`, consider reconfiguring rsyslog to only keep one rotation and rotate daily. This requires a patch to the rsyslog package.

Tune the filesystem

The default Lucid filesystem is ext4, and the default setting for ext4 is to use a 4K blocksize. With this setting, even creating a one-byte file will result in a 4K block allocation. On a system with many small files, this default blocksize will produce much wasted space.

ext4 can be configured to use a 1K block size, which tends to be more space-efficient on a small-disk system. The configuration must occur at partition and filesystem creation time, meaning you must modify the installer in order to pass the correct arguments to *mkfs.ext4*.

The *-T small* argument is perfect for tuning ext4 on a small 512M disk.

```
# mkfs.ext4 -T small /dev/sda1
```

To enable this option in your install image, modify your preseed configuration in the following fashion:

```
--- install/disk-recipe.orig
+++ install/disk-recipe
@@ -12,6 +12,7 @@
        format{ }
        use_filesystem{ }
        reserved_for_root{ 1 }
+       usage{ small }
```

```
        $default_filesystem{ }
        mountpoint{ / } .
```

Using this technique on an i386 test system reduced the disk footprint to 604M from a 736M footprint using 4K blocks.

More partman documentation can be found here:

- https://help.ubuntu.com/10.04/installation-guide/i386/preseed-contents.html
- http://d-i.alioth.debian.org/svn/debian-installer/installer/doc/devel/partman-auto-recipe.txt

However, please note that this severely increases boot time and decreases performance.

Explore alternate filesystems

The Linux world eagerly awaits the stabilization of btrfs. It promises ponies for everyone, including on-the-fly file compression. This feature gives btrfs a huge advantage over squashfs for small systems, since the filesystem is read-write versus read-only.

The Lucid kernel has the technical capability to use btrfs partitions, but it is completely unsupported.

That being said, it is indeed possible to convert your root filesystem to btrfs and boot into it with the default Lucid kernel.

One technique is described in this btrfs rootfs howto. While the basic premise described in the post is sound, it unfortunately does not demonstrate the capability of btrfs data compression, since the existing ext4 data are converted in-place, and are not packed as tightly as they could be otherwise.

One might think that using the *btrfsctl* utility to defragment the btrfs partition would be the solution, and conceptually, one would be correct. Unfortunately, the version of *btrfs-tools* in Lucid is not new enough. One cannot simply build the latest btrfs-tools from upstream either, since the userspace tools are tightly coupled with the kernel version.

A working technique is to leave some free space on the drive during installation instead of using the entire disk for the installation. After installation completes and the system reboots, boot into the live image. Create a new btrfs partition in the free space, and mount it with the *compress* option. Then, copy all the data from the ext4 partition to the btrfs partition: *cp -ax* is your friend.

You can now point grub at your btrfs partition using the technique described in the howto, and Lucid will boot quite happily.

Using this technique on an i386 test system reduced the disk footprint to 303M from a 736M ext4 footprint.

The btrfs wiki is a useful resource.

Reconfigure and rebuild the kernel

The default i386 Lucid kernel and modules occupy 86M of disk. This space can be reclaimed by only building the modules needed for your target platform.

Even worse, modules are double-counted, since they are included in the initrd as well. Consider building your kernel without an initrd. You'll not only save space, you'll also decrease the boot time as well. Bonus!

Device firmware also needlessly occupies disk space. Be sure to remove the firmware files unnecessary for your platform.

Remove unnecessary kernel modules

If you have not reconfigured the kernel, consider simply removing the unused modules and firmware from your system. Check which modules get loaded (plug in all necessary hardware first so that all drivers get loaded), and then remove all kernel modules from `/lib/modules/`*version* which are not listed in `lsmod`.

Obviously this technique is not easily maintainable.

If you are using live-helper, a hook like this can save you a lot of space.

Other potential possibilities

These do not have a readily made implementation yet, but are worth investigating if needed:

- Python modules are typically compiled into bytecode (*.pyc files) as a performance boost. However, they too occupy precious space and can be removed safely without harming the system. The next package update will regenerate those, though (through `pycentral` or `python-support`).

- Check `/usr/share/themes`, XFCE and GNOME provide quite a lot by default → provide only a subset

- /usr/share/i18n/locales → gzip [4 MB]
- drop unnecessary packages:
  - ubuntu-minimal: vim-common vim-tiny
  - should be removed by installation: redboot-tools os-prober parted ...
  - installer cruft: devio sgml-base aptitude dmsetup make gnupg-curl
  - pango: whiptail defoma xterm
- /usr/share/zoneinfo → gzip? (needs messing with hardlinks)

- `gzip -9 /usr/share/perl/5.10.1/unicore/*.txt` and fix perl to get along with it [3 MB]

ReducingDiskFootprint (last edited 2012-10-11 13:54:27 by pitti)