Secondary Arches, enabling Fedora to run everywhere

Dennis Gilmore
Fedora Project OLPC
dgilmore@fedoraproject.org

Abstract

There are many different types of CPU out there. From SPARC to Alpha, Arm to s390. Most people run i386 or x86_64; how can you get a distro for your favorite exotic preference? Fedora has been doing a lot of work to allow interested people to build and support the architecture of their choice. Work has been going on for arm, alpha, ia64, s390, and SPARC.

This paper will cover what has been done to enable motivated people to bring up and support new architectures. The same tools used to make Fedora can be used to layer products on top of Fedora also. If you're interested in MIPS, PA-RISC, or just want to know what is involved in putting together a distro, this paper is also for you.

1 What is a secondary architecture?

Any architecture that is not mainstream? Fringe architectures? Anything not x86 or x86_64?

Fedora is starting by saying that it is anything that is not x86 or x86_64 based, however ppc is currently a special case. We are building ppc and ppc64 with the primary architectures, however the release engineering for it is being done by the Fedora community. This is also not to say that the way things are done now is the way they will always be. For instance if one architecture started building up market share then it would be evaluated for primary architecture inclusion.

The secondary architecture team is responsible for providing all of the building and hosting resources. The team also gets full access to Fedora cvs. The same source cvs checkout will be used to build across all architectures.

Fedora package maintainers are encouraged to look at and fix build failures on secondary architectures. Howver, they are not required to do so. We understand that there are people who have no interest in supporting secondary architectures. The architecture teams are there to ensure things get fixed, and to provide architecture specific knowledge to interested maintainers.

We are actively looking for sponsorship of a master mirror for secondary architectures to reduce the barrier of entry for all. While the barrier of entry is high for a new Secondary Architecure it will ensure that the people proposing the architecture are commited to ensuring that it is a success.

The initial proposal was written by Tom Callaway with the following purpose. As an open community, Fedora encourages motivated individuals who care about architectures which are currently unsupported [1]. With the ultimate goal of supporting as many architectures as people want to support.

2 Why support secondary architectures?

The more architectures that you build the same code on, the more portable the code base. Lots of people make a claim that their code is portable because they write to standards, such as POSIX. Many times they really have no idea if their code is truely portable.

What is portable code? The same code running in different places. We are testing portability by building on big endian CPUs, little endian CPUs, CPUs with different instruction sets. We could also test portablity by building on different OS's such as FreeBSD, Darwin, or Hurd on the same CPU.

We also scratch people's itches. If you want to know more about Linux and how a distribution is put together, then working on a secondary architecture is a great place to start.

3 Who is doing secondary architectures?

- HP, Intel, SGI, and Red Hat for Itanium

- Red Hat and IBM for S390
- Marvell for ARM
- Fedora Community for SPARC
- Fedora Community for ALPHA

Lots of different parts of the Fedora community are working on actively ensuring the success of Secondary Architectures. Secondary architectures are not a new idea. Aurora SPARC Linux, Alpha Core, and RHEL have been doing it for years. This is a new process that simplifies and speeds up getting other architectures built.

4 How do Secondary Architectures work?

All packages will be built on the primary hub first. Once a package has been succesfully been built it will be queued on the secondary hub. There is a many-to-one relationship between primary and secondary hubs.

Initially all automated communication is one way. A long term goal is to push builds back up to the primary architecture hub. `kojisd` is a new tool that has been written to enable automatic builds on secondary hubs. `kojisd` ensures that the secondary hubs have the same or newer build of everything used in the buildroot as what was used on the primary hub. Architecture teams are responsible for ensuring that bugs get fixed, and doing the release enginering.

5 Architecture resources needed (provided by architecture team)

- 1TB disk space per architecture
  This will be an ever growing amount of disk. `koji` has a garbage collection mechanism to clean up old builds, however Fedora builds a huge number of packages, the number of builds will continue to grow as new packages are added. For Fedora-9 there were ~20000 builds.

- web server
  Used for the `koji` frontend and hub, running apache with mod_python, the hub uses xmlrpc for all communication. Fedora provides a SSL cert to be used for user authentication. This allows users to use the same credentials and work on any architecture.

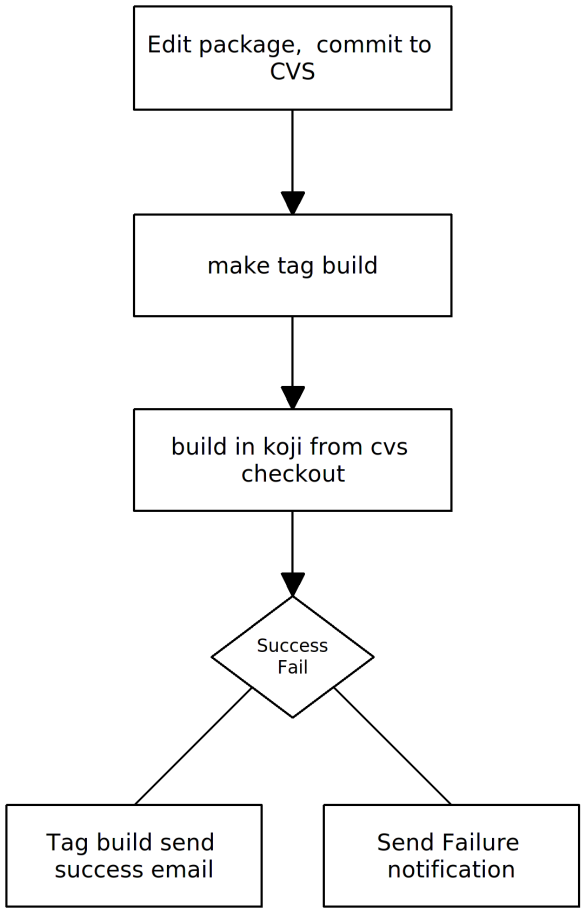

Figure 1: previous workflow

- Database server
  Postgresql server, depending on the hardware, this can run on the web server. Ideally this is a dedicated server. Some tables like the rpmfiles table contain over 90,000,000 rows in the Fedora installation. Fedora is able to provide scripts to enable backups of the database. Postgres 8.3 is recommended due to it having working autovacuum support.

- builders
  At least one but preferably more, they need to have read only access to the file system that koji stores its packages on, this can be via http, nfs, iscsi, etc. In order to run createrepo tasks, at least one builder needs the local access to the koji file system.

- bandwidth
  Pushing up images, rawhide, and updates takes up bandwidth, uploading 30 to 60 gigabytes for a release at 1 mbit/s takes a long time.

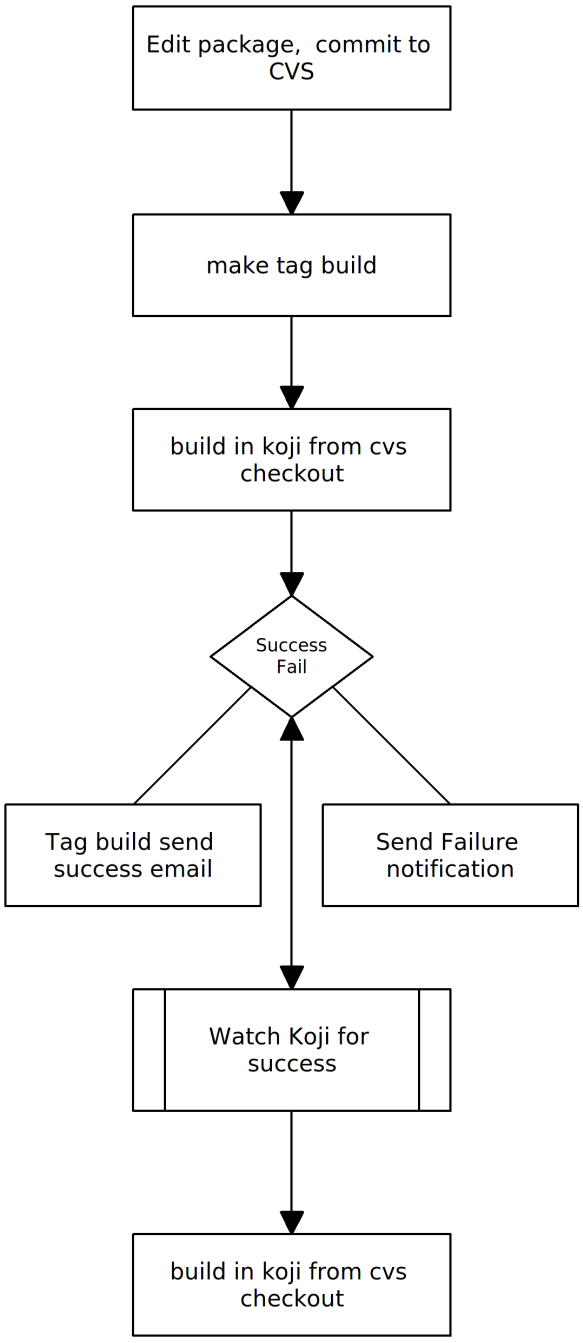

Figure 2: new workflow workflow

- mirrors
  Right now we are unable to provide mirrors. We are working on enabling hosting and mirroring of release trees. Secondary architectures will always be responsible for hosting their own infrastructure.

- Sand Box Machines (optional but recommended)
  Gives a workspace for Fedora maintainers to do test builds and debug failures using mock. This will be an environment that will help people who want to help but don't have access to hardware.

6 Fedora Provided Resources

- Package CVS
  Every package built for all architectures must come from Fedora cvs. This will ensure that we can be in compliance with the GPL, everyone will also know where to go.

- User authentication
  All users on secondary architecture hubs are identified by using the same certificates, there is seamless SSL authentication across all secondary architecture hubs.

- Packagers
  Fedora has 596 packagers maintaining over 6000 source packages. Many, but not all of these people, will help fix bugs on secondary arches for the packages they maintain.

- Framework
  There is an existing framework and mechanism to support new contributors and contributions. Package management, workflow management, and user support are all taken care of, greatly lowering the barrier to entry of starting your own distribution project. Fedora has resources for hosting upstream projects called fedorahosted. As well as VoIP and mailing lists for communications. Fedora uses freenode for irc communications.

- Best practice guidelines
  Fedora has a set of packaging guidelines `http://fedoraproject.org/wiki/Packaging/Guidelines` which are constantly evolving and growing as new types of software are added to package repositories. Fedora is at the forefront of Linux development. Many things like selinux, xen, gcj, OpenJDK and NetworkManager shipped first in Fedora. Fedora is also heavliy involved in upstream development of many technologies. New gnome releases regularly ship first in Fedora.

- Release engineering tools and support
  Fedora's insistance that everything be open means that not a single proprietary tool is used to produce the distribution. In the past the hardest part

in doing your own release was getting tools to do composes. With the merge of Core and Extras Red Hat's previous tools were dropped and new open source tools were created. Fedora Release Engineering is entirely done in the open, and release engineering team members are available to help with issues. As things move forward we will have new problems to tackle, being open we will be able to resolve them all.

7 Existing Fedora Tools

- koji
  `http://koji.fedorahosted.org`
  Terminology

  In Koji, it is sometimes necessary to distinguish between the package in general, a specific build of a package, and the various rpm files created by a build. When precision is needed, these terms should be interpreted as follows:

  Package:

  - The name of a source rpm. This refers to the package in general and not any particular build or subpackage. For example: kernel, glibc, etc.

  Build:

  - A particular build of a package. This refers to the entire build: all arches and subpackages. For example: kernel-2.6.9-34.EL, glibc-2.3.4-2.19.

  RPM:

  - A particular rpm. A specific arch and subpackage of a build. For example: kernel-2.6.9-34.EL.x86_64, kernel-devel-2.6.9-34.EL.s390, glibc-2.3.4-2.19.i686, glibc-common-2.3.4-2.19.ia64.

  Koji Components

  Koji is comprised of several components:

  - koji-hub is the center of all Koji operations. It is an XML-RPC server running under mod_python in Apache. koji-hub is passive in that it only receives XML-RPC calls and relies upon the build daemons and other components to initiate communication. koji-hub is the only component that has direct access to the database and is one of the two components that have write access to the file system.
  - kojid is the build daemon that runs on each of the build machines. Its primary responsibility is polling for incoming build requests and handling them accordingly. Koji also has support for tasks other than building. Creating install images is one example. kojid is responsible for handling these tasks as well. kojid uses mock for building. It also creates a fresh buildroot for every build. kojid is written in Python and communicates with koji-hub via XML-RPC.
  - koji-web is a set of scripts that run in mod_python and use the Cheetah templating engine to provide an web interface to Koji. koji-web exposes a lot of information and also provides a means for certain operations, such as cancelling builds.
  - koji is a CLI written in Python that provides many hooks into Koji. It allows the user to query much of the data as well as perform actions such as build initiation.
  - kojira is a daemon that keeps the build root repodata updated.

  Koji organizes packages using tags. In Koji a tag is roughly analogous to a beehive collection instance, but differ in a number of ways:

  - Tags are tracked in the database but not on disk.
  - Tags support multiple inheritance.
  - Each tag has its own list of valid packages (inheritable.)
  - Package ownership can be set per-tag (inheritable.)
  - Tag inheritance is more configurable.
  - When you build you specify a target rather than a tag.

  A build target specifies where a package should be built and how it should be tagged afterwards. This allows target names to remain fixed as tags change through releases [2].

- mash
  `http://mash.fedorahosted.org`

mash is a tool that creates repositories from koji tags, and solves them for multilib dependencies.

- pungi
  `http://pungi.fedorahosted.org`
  The pungi project is two things. First and foremost it is a free opensource tool to spin Fedora installation trees and isos. It will be used to produce Fedora releases from Fedora 7 on, until it is replaced by something better. Secondly, pungi is a set of python libraries to build various compose-like tools on top of. Pungi provides a library with various functions to find, depsolve, and gather packages into a given location. It provides a second library with various functions to run various Anaconda tools on the gathered packages and create isos from the results.

- livecd-tools
  livecd-tools lives in a git repo at `http://fedorahosted.org/`. In a nutshell, the livecd-creator program

  - Sets up a file for the ext3 file system that will contain all the data comprising the live CD.
  - Loopback mounts that file into the file system so there is an installation root.
  - Bind mounts certain kernel file systems (/dev, /dev/pts, /proc, /sys, /selinux) inside the installation root.
  - Uses a configuration file to define the requested packages and default configuration options. The format of this file is the same as is used for installing a system via kickstart.
  - Installs, using yum, the requested packages into the installation using the given repositories.
  - Optionally runs scripts as specified by the live CD configuration file.
  - Relabels the entire installation root (for SELinux.)
  - Creates a live CD specific initramfs that matches the installed kernel.
  - Unmounts the kernel file systems mounted inside the installation root.
  - Unmounts the installation root.
  - Creates a squashfs file system containing only the ext3 file (compression.)
  - Configures the boot loader.
  - Creates an iso9660 bootable CD.

8 New Tools Needed

- kojisd
  kojisd is a new tool written to monitor one hub and on success of tasks to repeat the task on a second hub. There are two things that we want to do. Maintain tags/targets and build packages using a buildroot which is as identical as possible. Due to the high possibility during rampup that packages will fail we are saying that if we have a newer build on the secondary hub then that's ok. It also simplifies buildroot management. We can follow the same principles that the primary hub does. The one thing we add is to ensure that for instance things depending on a specific version of xulrunner are built against that version since build timings will vary on different architectures.

  Many options in kojisd are configurable. For instance you can choose to import packages based on architecture, and whitelist/blacklist targets to build for. For Secondary Architecture purposes we will be importing noarch packages. For someone layering products on top of Fedora they can import all builds.

9 Secondary architecture Teams

We currently have teams working on ARM, Alpha, Itanium, S390, and SPARC.

9.1 ARM

- Hub: Not yet available.
- Team Page: `http://fedoraproject.org/wiki/Architectures/Arm`

Arm is being lead by Lennert Buytenhek. The baseline ARM CPU architecture that we have chosen to support is ARMv5, Little Endian, Soft-Float, EABI. We believe that this provides a nice baseline and that the prebuilt packages and root file system images will be usable on many of the modern ARM CPUs, including XScale, ARM926, and ARM-11, etc.

9.2 Alpha

- Hub: `http://alpha.koji.fedoraproject.org`
- Team Page: `http://fedoraproject.org/wiki/Architectures/Alpha`

The Alpha team is being lead by Oliver Falk and has Jay Estabrook as a member. Fedora Alpha started out as the AlphaCore Linux Project, when Red Hat stopped support for Alpha with Red Hat Linux 7.1. Fedora Alpha is the continuation of the AlphaCore efforts, in an official capacity as part of the Fedora Project.

9.3 Itanium

- Hub: `http://ia64.koji.fedoraproject.org`
- Team Page: `http://fedoraproject.org/wiki/Architectures/IA64`

The IA64 team is being lead by Doug Chapman, and has Tim Yamin, Prarit Bhargava, Yi Zhan, Jes Sorensen, and George Beshers as members. The IA64 work is based on unofficial releases composed from builds as a side effect of Fedora's buildsystem before the merge. It was maintained in sorts after and is now becoming official.

9.4 S390

- Hub: `http://s390.koji.fedoraproject.org`
- Team Page: `http://fedoraproject.org/wiki/Architectures/s390x`

The S390 team is being lead by Brad Hinson and has Brock Organ as a member. The work on s390 is being based on RHEL5.

9.5 SPARC

- Hub: `http://sparc.koji.fedoraproject.org`
- Team Page: `http://fedoraproject.org/wiki/Architectures/SPARC`

The SPARC team is lead by Tom Callaway, and has Peter Jones, Patrick Laughton, and I as members. We have a wide range of SPARC hardware. Fedora, unlike Aurora SPARC Linux, will support only UltraSPARC and higher hardware due to lack of an upsteam kernel maintainer for older SPARC. All user land is being built for sparcv9 and sparc64, with sparcv9 recommended for use.

10 How can the Secondary Architecture work help layering products on top of Fedora?

Using the tools written for secondary architectures, you are able to pull into your own koji setup builds from Fedora as they happen. Using a trigger mechanisim you are able to build your own packages when new packages are built in Fedora.

References

[1] Fedora Secondary Architecure proposal, `http://fedoraproject.org/wiki/TomCallaway/SecondaryArchitectures`.

[2] Fedora wiki Describing koji. `http://fedoraproject.org/wiki/Koji`.