ROYAL INSTITUTE
OF TECHNOLOGY

Efficient and Reliable Filesystem Snapshot Distribution

LAURI VÕSANDI

Master of Science Thesis in
Information and Communication Technology
Supervisor: Lars Kroll
Examiner: Dr. Jim Dowling
Stockholm, Sweden, June 2015

TRITA-ICT-EX-2015:99

Abstract

Linux is an portable operating system kernel devised by Linus Torvalds and it can be used in conjunction with other userspace utilities such as GNU to build a free and open-source operating system for a multitude of target applications. While Linux-based operating systems have made significant progress on the servers and embedded systems, there is still much room for improvement for workstations and laptops.

Up to now Linux-based operating system deployment has been error prone, time-consuming process and usually specific to a particular distribution of Linux. Linux-based operating systems also have a reputation of being overly complex to set up for a novice computer user and even though there are now laptops available with pre-installed Ubuntu [1], installing Linux-based operating system on arbitrary device is troublesome due to lack of native support for certain hardware components.

In this thesis Butterknife, a B-tree file system (Btrfs) and Linux Containers (LXC) based provisioning suite is presented. Butterknife can be used to significantly reduce deployment time of customized Linux-based operating system. Butterknife makes use of LXC to prepare a template of the root filesystem and Btrfs snapshotting to save state of the template. Btrfs send/receive mechanism is then used to transfer the root filesystem to the target machine. Post-deployment scripts are then used to configure the root filesystem for particular deployment, optionally retaining hostname, domain membership, configuration management keys etc. Current implementation of Butterknife uses HTTP(S) and multicast for transport, and various peer-to-peer scenarios are discussed in the Section 6 – Conclusions and Future Work.

In addition to provisioning, Butterknife makes use of Btrfs incremental snapshots to implement differential upgrades. This approach is especially attractive for mobile devices, embedded systems and Internet of Things, where software upgrades have to be delivered in a guaranteed manner. Butterknife brings additional value to already existing ecosystem by bridging gap between stock installation medium and configuration management.

Acknowledgements

I would like to thank my examiner *Dr. Jim Dowling* and my supervisor *Lars Kroll* firstly for providing me with the opportunity to carry out this research and secondly for guiding me through it, as well.

Furthermore, I wish to thank *Kalle Kebbinau* for providing a constant stream of valuable input and feedback on my work and ideas. I'd like to thank Btrfs community for merging the patches and being helpful overall.

I'd also like to thank international Free and Open-Source Software community for the immense pool of freely available components that made it possible to learn and build anything imaginable.

Finally I would like to thank Skype for funding my studies during first year at Technical University of Berlin and Archimedes Foundation for providing Kristjan Jaak Scholarship throughout the second year at Royal Institute of Technology.

Most of all I'd like to thank *Peeter Laanoja*, *Stanislav Kuhtinski*, *Alan Ôis*, *Erko Valdmets*, *Silver Püvi* and others participants of the Tallinn Education Department pilot project for their patience and feedback.

Contents

Chapter 1

Introduction

Linux is an operating system kernel developed by Linus Torvalds [2] and initially released in 1991. Linux in conjunction with userspace utilities such as GNU [3] is known as GNU/Linux. Linux-based operating systems are widely used in servers. Linux-based Android exceeded 85% smartphone market share in 2014 [4]. Even though Ubuntu has managed to gain some market share Windows continues to be dominant operating system for desktops and laptops next to steadily increasing Mac OS X [5].

Up to now Linux-based operating system deployment has been error prone, time-consuming process and usually specific to a particular distribution of Linux. Linux-based operating systems also have a long history of being overly complex to set up for a novice computer user. Even though Linux-based operating systems have made significant progress on the servers [1] and embeddeded systems [2], the workstations and laptops have been left without attention. In this paper technical solution to quickly deploy Linux-based operating systems and update software using Btrfs filesystem snapshots is presented.

1.1 Motivation

The foundation of current work was established while author was setting up the infrastructure to deploy Ubuntu 12.04 LTS on the PC-s of educational institutions of Tallinn as part of the ongoing efforts of Tallinn Education Department to switch from proprietary tools to open solutions in order to avoid vendor lock-in.

Puppet was set up to manage Ubuntu workstations remotely. Local IT-support took the role of bootstrapping the machines and joining them to remote management server.

[1] Google and Amazon have successfully used open-source software and commodity hardware to build enterprise grade cloud

[2] Android is the market leader in smartphones and it is composed of open-source components such as Linux

Customer A runs hundreds of embedded ARM computers for digital signage. Software is currently updated by mailing the customer an SD-card with updated software. The customer would prefer to update software and media over the air but the software update atomicity has to be guaranteed in order to avoid non-booting machines.

Customer B is about to deploy thousands of Ubuntu netbooks to be used as remote workstations around the globe. It is vital to unroll security updates as soon as possible, but at the same time it's necessary to guarantee software update atomicity as the IT helpdesk is lacking in the remote locations where the machines are used. The solution has to be installable at customer premises and it must make use of standard and recognized security methods.

Customer C has around thousand PC-s that need to be converted to Ubuntu, but the budget is lacking and therefore manual labour has to be minimized. Glitch-free software update mechanism is crucial part of minimizing manual labour.

1.2 Contributions

The work produced a novel method of deploying and maintaining Linux based workstations in an guaranteed and secure manner.

- Method for preparing template root filesystems using LXC.
- Method for deploying templates using Btrfs send/receive.
- Method for updating machines using Btrfs incremental updates.

1.3 Related Work

1.3.1 Puppet and Foreman

Puppet is a remote management system which features its own declarative domain-specific language to describe the state of the configuration [6]. Puppet server also known as Puppetmaster hosts the configuration while managed machines run puppet agent which polls the puppetmaster at specified interval, usually 30 minutes. Taken actions are then reported back to the puppetmaster. Puppet agent and puppetmaster both are written in Ruby and released the latest versions are released under liberal Apache 2.0 license. Puppet can be used to manage both Linux and Windows servers and workstations as well.

Puppet uses TLS based security model: Puppetmaster maintains a certificate authority which is used to sign certificate requests submitted by clients. Fully qualified hostname is used to identify nodes and it is also used derive common name for the X509 certificates. Once a certificate is signed the nodes are expected to execute whatever Puppetmaster dictates them to do.

Foreman is a complete lifecycle management software for physical and virtual servers. Foreman incorporates Puppet, a custom web interface and provisioning

tools into single unified application. Even without using provisioning features Foreman makes one of the most feature-complete web interfaces for Puppet rivalling the Puppet Dashboard.

1.3.2 Chef, Ansible and Salt

Chef is infrastructure automation tool. Chef is written in Ruby and Erlang. Chef uses domain-specific language written in Ruby. Chef server stores your recipes as well as other configuration data. The Chef client is installed on each server, virtual machine, container or networking device or generally speaking node. The client periodically polls Chef server latest policy and state of the network. If anything on the node is out of date, the client brings it up to date [7].

Ansible remote management software uses Secure Shell (SSH) to connect to the nodes which means there is no agent running on the managed machine, this however makes it slightly more complicated to use Ansible to manage machines behind NAT. Ansible is written in Python and it uses state-driven resource written in YAML.

Salt is an open-source configuration management system, capable of maintaining remote nodes in defined states and a distributed remote execution system used to execute commands on remote nodes, either individually or by arbitrary selection criteria [8]. Salt is developed by SaltStack which sells services around Salt. Salt uses ZeroMQ to transfer data between Salt server and Salt minions. Currently alternative transport RAET (Reliable Asynchronous Event Transport) is in development and it is designed Salt in mind. RAET attempts to address more complex message routing schemes which are not possible with ZeroMQ [9].

1.3.3 Fully Automated Installation

Fully Automated Installation (FAI) is a non-interactive system to install, customize and manage Linux systems and software configurations on computers as well as virtual machines and *chroot* environments, from small networks to large-scale infrastructures like clusters and cloud environments. It's a tool for unattended mass deployment of Linux. The systems are installed, and completely configured to your exact needs, without any interaction necessary [10].

1.3.4 Clonezilla, Symantec Ghost, Acronis True Image

Clonezilla [11] combines various open-source tools into a single cloning suite. Clonezilla uses partclone utilities [12] to identify and transfer only used blocks of various filesystems, most notably NTFS, Ext4 and Btrfs. Clonezilla supports resizing filesystems after the clone in order to make use of the whole disk space available in the target machine. This makes it possible to use same prepared image for disks of various size, the template image has to be of course smaller than the target machine disks. Clonezilla supports most Windows and Linux filesystems.

Symantec Ghost, previously known as Norton Ghost is a corresponding commercial product currently available on the market [13] offered by Symantec Corporation.

Acronis True Image [14] by Acronis International GmbH is another similar product which supports Windows and Mac OS X operating systems.

The fact that machines need to be taken offline is the main drawback of classic disk cloning methods.

1.3.5 FSArchiver

FSArchiver is a tool very much similar to Clonezilla, but instead of storing disk image on a block level the contents are stored on object-level (file, directory). All filesystem attributes are preserved for Linux filesystems, NTFS support is still experimental. For archiving the filesystem has to be unmounted or mounted read-only. Read-write mounted filesystems can be snapshotted with the assistance of LVM and archived afterwards [15].

1.3.6 BSD Jails, Solaris Zones, OpenVZ, Linux Containers, systemd-nspawn

Jails have been available in FreeBSD since version 4.x. Jails use *chroot syscall* to substitute root filesystem of a process making it possible to create a restricted environment which is isolated from the rest of the operating system [16].

Solaris Zones were introduced few years later adding similar capabilities to Solaris operating system. Solaris Zones took advantage of ZFS filesystem making it possible to snapshot and clone zones.

Linux has included *chroot* for long time as it's essential feature for switching from initial root filesystem (*initramfs*/*initrd*) to actual root filesystem. Many network services take advantage of *chroot syscall* to confine itself to a particular directory in order to mitigate consequences of vulnerabilities and exploits.

The main issue with *chroot* is that dependencies of the target application have to be available in the *chroot* root filesystem. For instance a Python application which has modules loaded before *chroot* operation could operate without any files in the *chroot*, but shell script which relies on several executables need to have those utilities available in *chroot* as well. With copy-on-write and de-duplicating filesystems such as Btrfs and ZFS the problem how ever becomes irrelevant as root filesystem of the *chroot* can be duplicated with no significant overhead.

Linux Containers (LXC) [17] takes advantage of the *chroot syscall* and recently Linux *cgroups* (control groups subsystem) which permit more operating system level virtualization. Control groups are used to implement limiting, accounting and isolation of CPU, memory, disk I/O, network, etc resource usage. LXC allows various backing stores, most notably ZFS, Btrfs and OverlayFS which make it very easy to enable container snapshotting and streaming backups.

Controversial *systemd* now includes `systemd-nspawn` which can be used to start up a container using kernel namespaces very much like LXC does. Another utility `machinectl`, bundled with *systemd* can be used to manage the containers [18].

1.3.7 Docker and Rocket

Docker started off as a way to automate container deployment and configuration using containers and control groups present in Linux kernel. As Docker started to add features that CoreOS developers deemed excessive an alternative project Rocket was founded [19]. Rocket aims As Amazon also announced EC2 Container Service, Rockets plan to formalize the container standard makes it possible have various implementations while maintaining cross-vendor compatibility.

1.3.8 CoreOS and Ubuntu Core

CoreOS [20] is a rearchitected Linux distribution which provides minimalist foundation to run Docker containers. It uses two-partition scheme to provide atomic updates of the root filesystem. The operating system runs off a read-only filesystem while the other one can be patched runtime. Reboot or *kexec* can be used to boot into the updated system. This prevents rendering device unbootable due to interrupted upgrade.

Ubuntu Core is an Ubuntu flavour tailored towards Internet of Things and as a container platform. Ubuntu Core introduced root filesystem transactional updates to Ubuntu using Snappy [21]. Ubuntu Core is designed to run Docker applications. Snappy is also plays important role in Ubuntu Phone ecosystem. Snappy uses dual-partition scheme – one of the root filesystems is mounted read-only and writable directories are mounted from writeable filesystem.

1.3.9 OverlayFS

OverlayFS is a feature introduced in Linux 3.18 which makes it possible to merge contents of two separate mountpoints on the fly [22]. OpenWrt uses OverlayFS to implement writable JFFS2 layer on top of read-only SquashFS filesystem.

1.3.10 LVM, mdadm and dmraid

Ext4 has been primary filesystem for Linux based workstations and servers for a while. It provides filesystem primitives such as files, directories, permissions and timestamping. In order to add redundancy either software Redundant Array of Independent Disks (RAID) or Logical Volume Management (LVM) can be used.

Software RAID is implemented in Linux by means of *mdadm*. Software RAID can be used to build RAID1, RAID0, RAID10/01, RAID5 or RAID6 arrays without dedicated RAID controller which could also impose a vendor lock-in.

LVM enables pooling of drives, mirroring and snapshotting by adding an abstraction layer on top of physical disks. Any filesystem that can be deployed on physical disk can also be deployed on top of LVM's logical volume. The kernel takes care of mapping logical addresses to corresponding disk's physical address. The snapshotting feature of LVM however has been claimed to be buggy. [23]

1.3.11 Btrfs and ZFS

Btrfs and Zettabyte Filesystem (ZFS) both are modern copy on write filesystems which also fill in the role of volume manager [24] [25]. Btrfs has been claimed to be unstable but the situation has improved significantly over the past year or two. Facebook has been testing Btrfs in production since the April of 2014. [26] Chris Mason, a lead developer of Btrfs joined Facebook in the end of 2013 with the goal of improving Btrfs support for enterprise applications. [27]

Btrfs supports redundancy in RAID0/1/10 configurations and RAID5/6 support was added with Linux 3.19 [28]. Btrfs supports *zlib* and *lzo* compression algorithms, however enabling compression for Btrfs is known to seriously hamper performance of database engines.

Btrfs supports subvolumes, which makes it possible to group directories and files into logical units. As mentioned above, LXC makes use of subvolumes by confining the root filesystem to a Btrfs subvolume if Btrfs backing store is used. This means that root filesystem of a container can easily be snapshotted.

In Btrfs context snapshotting is actually a generalization of subvolume cloning, thus *subvolume*, *clone* and *snapshot* may be used interchangeably in certain contexts. Due to copy-on-write architecture subsequent writes to the original subvolume do not affect the subvolume clone and there is no performance degradation for the original subvolume.

Btrfs permits transmitting filesystem data over the network by a concept of `btrfs send` and `btrfs receive`. Btrfs send mandates the subvolume to be in a read-only mode, which means that `btrfs subvol snapshot` has to be issued with extra *-r* flag.

Btrfs uses *received_uuid* and *uuid* to identify snapshots for transfer. The *received_uuid* corresponds to the *uuid* on the initial machine and it remains intact for any subsequent transfers. For differential snapshots few extra steps are taken. During snapshot send an optimal parent snapshot is identified and that is used as basis for the differential snapshot. On the receiving end the a read-write snapshot of the parent snapshot is created and filesystem operations are replayed on the subvolume. Finally corresponding *received_uuid* is set and system call is issued to set subvolume read-only.

Debian introduced Btrfs with Squeeze and has improved support since then [29]. Ubuntu also has supported Btrfs for a while with customized subvolume naming scheme: root filesystem is placed in subvolume named @ and home directories in a subvolume called *@home*.

1.3.12 apt-btrfs-snapshot and yum-fs-snapshot

In Ubuntu package repositories there is available `apt-btrfs-snapshot` [30] package, which creates snapshot of the root filesystem subvolume @ before every `apt-get` operation.

`yum-plugin-fs-snapshot` [31] is the corresponding package for Fedora and Red Hat

based distributions.

These approaches make it possible to boot into previous snapshots in case there are issues with the updated packages.

1.3.13 FreeNAS, Rockstor and OpenMediaVault

FreeNAS is a FreeBSD [32] based distribution which builds a complete NAS solution on top of ZFS filesystem and web interface. Rockstor is a complementary CentOS 7 based solution that uses Btrfs instead of ZFS filesystem [33]. OpenMediaVault provides similar functionality using Debian Wheezy instead of CentOS [34]. All three of them support SMB/CIFS (Windows file shares), NFS (UNIX file shares) and filesystem aided snapshots.

1.3.14 rsync and rsnapshot

rsync is an open source utility designed for fast incremental file transfer. It is most commonly invoked with archiving flag (-a) which retains permissions, ownership, timestamps and symlinks. In that case files are transferred only if modified timestamp or file length differs.

rsnapshot is an utility that takes advantage of hardlinking functionality of Ext4 and other similar filesystems. By creating a clone of a directory tree using hardlinks no extra disk space is consumed. Applying classical *rsync* on top of the clone only increments by the disk usage of changed files.

1.3.15 OpenWrt and DD-WRT

Most Linux based open-source router firmwares originate from Linksys WRT54G wireless router. Initially Cisco did not provide source code for the router as GPL mandates. Between 2003-2008 Free Software Foundation attempted to cooperate with Cisco to work out issues, but as Cisco continued to release new devices with similar issues Free Software Foundation eventually sued Cisco for malpractice. After the lawsuit Cisco complied and has been releasing firmware sources for all of their devices which make use of software released under GPL licenses [35]. Linksys WRT54G sources were basis for various Linux distributions for routers such as DD-WRT [36], Tomato [37] OpenWrt [38].

OpenWrt as we know it today is a Linux distribution for embedded devices which attempts to provide writeable filesystem and package management. OpenWrt supported hardware list mainly targets routers, but other devices are listed as well [39]. OpenWrt can be used to extend lifetime of equipment that otherwise would be largely obsolete due to unmaintained software from hardware manufacturer.

Most high-end consumer grade routers employ 8MB NOR Flash chip which is directly connected to the SoC without controller [40]. The Flash storage is usually partitioned at least as 3 slices: bootloader, read-only root filesystem, read-write overlay.

The read-only root filesystem contains SquashFS [41] which is highly-efficient compressed read-only filesystem that supports variety of compression algorithms. The read-write overlay partition is formatted as JFFS2 (journalling flash filesystem).

This method makes it possible to: perform factory reset simply by formatting the JFFS2 partition and upgrading firmware by overwriting SquashFS partition. Due to lack of redundancy in consumer-grade routers an interrupted firmware upgrade usually renders device unfit for use [42] [43]. This is also known as *bricking* in embedded developer jargon.

1.3.16 Android

Android ROM images are typically distributed as zip files which contain binary blobs for modem and bootloader in addition to snapshots of the file primary filesystems of Android: *boot*, *cache*, *recovery*, *system* and *userdata* [3]. Differential images are also available, in that case zip file contains directory tree of files intended to be overwritten or added to the original root filesystem and post-installation scripts which correct the file permissions [4].

ROM manager such as ClockworkMod [44] or TWRP [45] has to be used to install or patch third-party ROM-s. An alternative Fastboot method is also present in most Android devices and it can be used to directly write raw filesystem images and unlock the device [46]. CyanogenMod is an aftermarket software for phones that are not supported by the manufacturer any more and it can be installed with TWRP or ClockworkMod. ClockworkMod and TWRP both can be used to download images in the zip format and then automatically reboot to the recovery mode to install the ROM or updates. Similar methods are provided by some of the hardware vendors to update the ROM over the air. As Android is open-source vendors tend to customize various aspects of the software update process, there is no single canonical way to update Android devices.

1.3.17 OpenStack

OpenStack is a free and open-source cloud computing software platform which is composed of several components of which most noteworthy for current work are: Glance image service, Ironic bare metal provisioning, Swift object storage and Cinder block storage. Glance provides discovery, registery and retrieval services of virtual machine images [47]. Glance BitTorrent delivery enables BitTorrent support for transferring the images [48].

Glance supports variety of disk formats [49]:

- VHD disk format, a common disk format used by virtual machine monitors from VMWare, Xen, Microsoft, VirtualBox, and others
- VMDK disk format supported by many common virtual machine monitors

[3] https://dl.google.com/dl/android/aosp/shamu-lrx22c-factory-ff173fc6.tgz
[4] http://gapps.itvends.com/gapps-lp-20141212-signed.zip

- VDI disk format supported by VirtualBox virtual machine monitor and the QEMU emulator
- ISO archive format used by optical disks eg CD-ROM
- QCOW2 disk format supported by the QEMU emulator that can expand dynamically and supports Copy on Write
- AMI disk format used for Amazon machine images.

As of February 2015 Glance work has started to add Open Virtual Appliance (OVA) and Open Virtualization Format (OVF) [50] support. The Glance API does not however address versioning of disk and container images, therefore differential updates of images are also not implemented.

Chapter 2

Background

2.1 Initial task

The initial task was to use Ubuntu as operating system basis for schools due to rich set of both open-source and proprietary software components available in Ubuntu ecosystem.

So far the machine deployment has been a tedious task involving manual labour: The Ubuntu 14.04 LTS image had to be downloaded from the Internet and transferred to a memory stick. The Ubuntu installer was booted from the memory stick and usual installation was performed which took roughly 20 minutes. The machine was booted into Ubuntu, Puppet was installed on the machine and Puppet configuration was tweaked to use our server. This took another 10 minutes and was not a procedure that could be performed by a novice user using (pseudo-)graphical user interface. Once the certificates was signed on the Puppetmaster the machine downloaded necessary packages and applied configuration changes. Usually this would take several Puppet runs and as a result setting up a classroom of computers took several days depending on the command-line proficiency of local IT-support, which in some cases was close to zero.

In earlier phases of the project customized Estobuntu [51] – remastered [52] Ubuntu 12.04 was used to bootstrap the machines, but as the packages were constantly updated in the Ubuntu repositories, the CD generation process was complex, time-consuming and error-prone this approach was eventually given up and vanilla [53] Ubuntu was used instead.

2.2 Problems with package management

Ubuntu uses APT (Advanced Packaging Tool) as basis for it's package management. APT was originally developed as part of Debian operating system to be used as *dpkg* front-end. While *dpkg* can be used to install and remove packages, it does not provide dependency tracking nor fetching packages from remote locations which are implemented by APT. APT significantly simplifies the installation of software

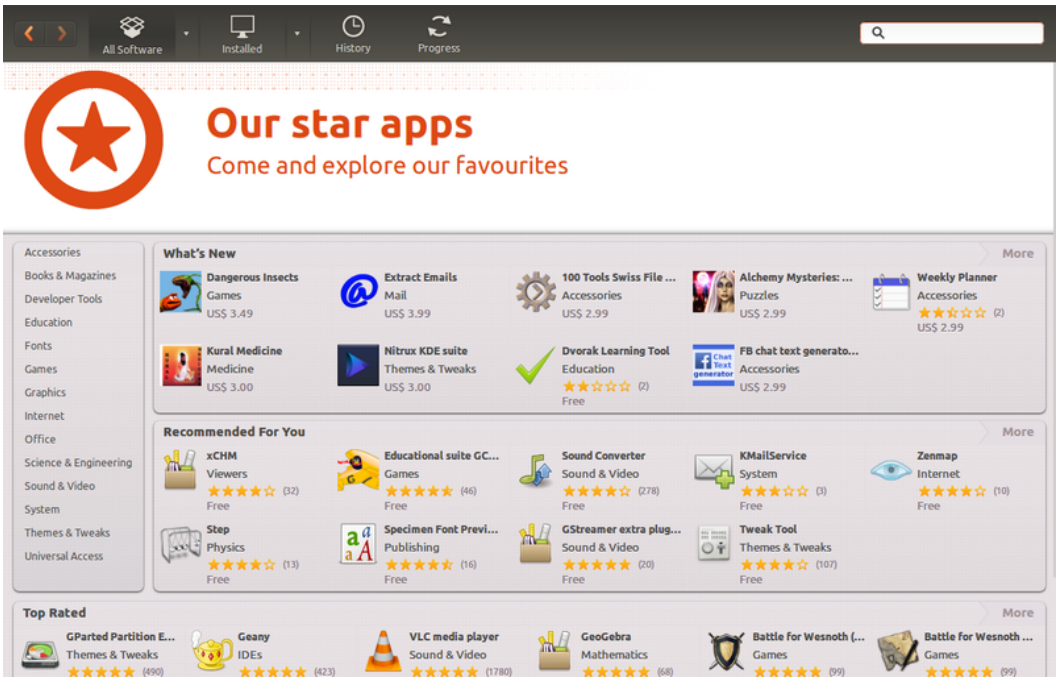

Figure 2.1. Ubuntu software center.

components by downloading packages from different sources and checking package dependencies prior installation.

Ubuntu Software Center shown in Figure 2.1 builds another abstraction on top of APT, while hiding libraries and other system components it enables even more simplified installation of applications for Ubuntu based machines. For remotely managed machines the Ubuntu Software Center and other graphical package management tools were removed.

There are however certain corner-cases where APT may render the package management unusable. For instance package list corruption was faced on several occasions, in that case APT crashes with segmentation fault [54] and currently the only known solution to the problem involves deleting package lists and running `apt -get update` again. Several faulty packaging scripts were stumbled on, for example it was not possible to remove certain versions of LibreOffice packages and manual intervention was necessary [55].

Debian community has been working hard to provide differential updates for the packages, but as of February 2015 the efforts have proven fruitless. Differential updates are applied for package lists [56], but binary diffs for packages have not implemented yet. Fedora community has however successfully deployed differential packages [57], thus reducing the amount of data needed to be transferred during an package update. For bigger software (eg LibreOffice) the lack of differential updates poses a serious concern, especially for low-bandwidth links. Puppet, SaltStack,

Chef, Ansible and other traditional configuration management fit best the scenario where each node has slightly different configuration and it makes sense to keep them separate. However provisioning very similar nodes with for instance Foreman has obvious overhead – each node has to fetch updated packages independently from the same APT repositories, same has to be done for application software.

Release upgrades for example from Ubuntu 12.04 to Ubuntu 14.04 have proven to be especially troublesome due to the fact that system libraries and files are being replaced and interrupted release upgrade may leave system in an unusable state.

As it has hopefully become clear by now installation of software for Ubuntu and Debian is most usually performed using APT in one form or another. Software that is not available in an APT repository is troublesome to install. For instance Smarttech distributes software for their smart whiteboard products as a .zip of Debian packages. Similarily Canon printer drivers are available as a .zip file. Last version of Acrobat Reader is also not available from any APT repository for Ubuntu 14.04. Skype distributes a Debian package for Ubuntu, but again not from a APT repository. Setting up an APT repository is not a trivial task, even for an experienced Linux sysadmin.

For classroom deployment cloning has been used in the past: Windows, Ubuntu or both are installed on a physical template machine. Template machine is thoroughly tested. Tools such as Clonezilla [11] or Symantec Ghost are used to transfer the hard disk image to the other machines. As the whole procedure is a complex undertaking it is usually performed once a year in summer especially for educational institutions. In fact cloning was used by some of the participating schools – for example Alan Õis, the IT-support at Mustamäe Gümnaasium used Clonezilla to set up his classrooms.

2.3 Specification

Considering the needs of the commercial customers and experience gained in the first iteration of the migration project following list of requirements were derived for next iteration.

Firstly the solution has to support all major Linux distributions – Ubuntu, Fedora, Red Hat, etc. The software upgrades have to be atomic, in other words interrupted updates can not render a workstation unusable. Software upgrades must retain domain join without having to join machine to a domain. The home directories must remain intact during software updates. It has to be possible to perform provisioning stage from Preboot eXecution Environment (PXE), off bootable USB memory sticks and optionally CD-R discs.

The HTTP(S) server has to be flexible enough to allow two main usecases – running a central server for all nodes and running a local downstream server, hence pushing and pulling templates between servers has to possible.

Following requirements were specified for security: As initial provisioning can be assumed to be done on premises, man-in-the-middle attacks can be ruled out

in that provisioning stage. It has to be possible to verify subsequent incremental snapshots by means of asymmetric keys. Consistent methods for root filesystem template fingerprinting have to be provided

Chapter 3

Butterknife Design and Architecture

We have outlined a number of existing methods and tools which address various aspect of Linux deployment on workstations in chapter 1, and it has hopefully become clear the functionality and guarantees provided by the presented systems. However, it is clear that there will never be a single tool that is optimal for every application.

Yet, many problems are a shared concern between all of these systems. Foremost among these are *bootstrapping*, i.e. getting the initial software setup on the machine, *upgrading*, that is updating the software components on the machine and *configuring* the software components.

To address this issue we present *Butterknife*, a provisioning suite that provides solutions for bootstrapping and upgrading Linux-based workstations, while remaining flexible enough to be used to deploy any Linux-based operating system and to be used in conjunction with already existing configuration management tools such as Puppet and Salt. Butterknife brings additional value to already existing ecosystem by bridging gap between stock installation medium and configuration management.

3.1 Concepts

The prototype draws inspiration from embedded computers where certain guarantees have to be provided [58]. Butterknife in its current form makes use of Btrfs filesystem and LXC. Current architecture of Butterknife provides atomic updates of root filesystem by making use of Btrfs snapshotting. Common filesystems such as ext2/3/4 were evaluated, but ruled out due to lack of snapshotting and send/receive functionality. ZFS was also considered, but ruled out due to licensing issues, memory usage and stability on 32-bit systems. Dual-partitioning schemes were considered, but ruled out due to disk usage overhead. CoreOS, Rocket, Ubuntu Core were also evaluated as viable platforms, but because of the way container virtualization technology is used in these platforms it's tricky to gain access to keyboard, mouse and video output from the container in addition to reliance on the OS vendor. Hence Butterknife simply uses containers to bootstrap the template, but containers

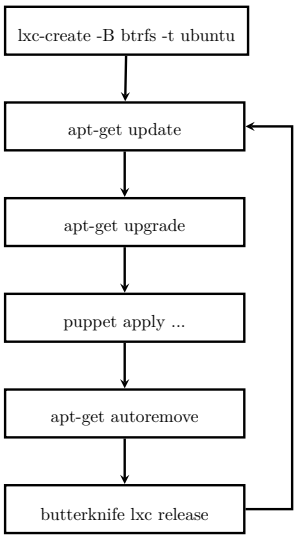

Figure 3.1. Template preparation workflow using Butterknife.

are not used on the target machine.

The template preparation workflow for Ubuntu based distribution and Puppet based configuration management is shown in Figure 3.1: LXC is used to bootstrap a container, `apt-get update` is used to update package lists, `puppet apply` is used to apply configuration on the container, `apt-get upgrade` is used to upgrade the packages and `apt-get autoremove` is used to remove packages that were installed as dependencies for packages removed by Puppet. Finally `butterknife lxc release` is used to stop the container and create a snapshot of the root filesystem of the container. For subsequent releases the cycle starts again with `apt-get update`. The Butterknife does not confine user to Puppet or Ubuntu, LXC supports a variety of distributions and scripts for Salt persistence are included in the template overlay directory of Butterknife Git repository.

The deployment workflow of the prototype is split into bootstrap and live stages as shown in Figure 3.2. The template prepared in the template preparation workflow is served by `butterknife serve`. Provisioning image is booted either via Preboot eXecution Environment (PXE) or from USB memory stick. Provisioning utility partitions the target machine storage device, creates Btrfs filesystem and transfers full snapshot of the template from Butterknife server. Target subvolume is mounted at `/mnt/target` and post-installation tasks, for example bootloader installation, are performed.

Finally the provisioning utility reboots the machine and system boots from the newly received subvolume. Butterknife DBus service starts up and connects to the Butterknife server waiting for subsequent snapshots. Once new snapshots are available, DBus service fetches the differential snapshots and adds them to the Btrfs pool. Snapshot is optionally verified and post-installation scripts are executed. User

is notified about requirement to reboot the machine to new snapshot via the DBus notification service. For subsequent snapshots the cycle repeats.

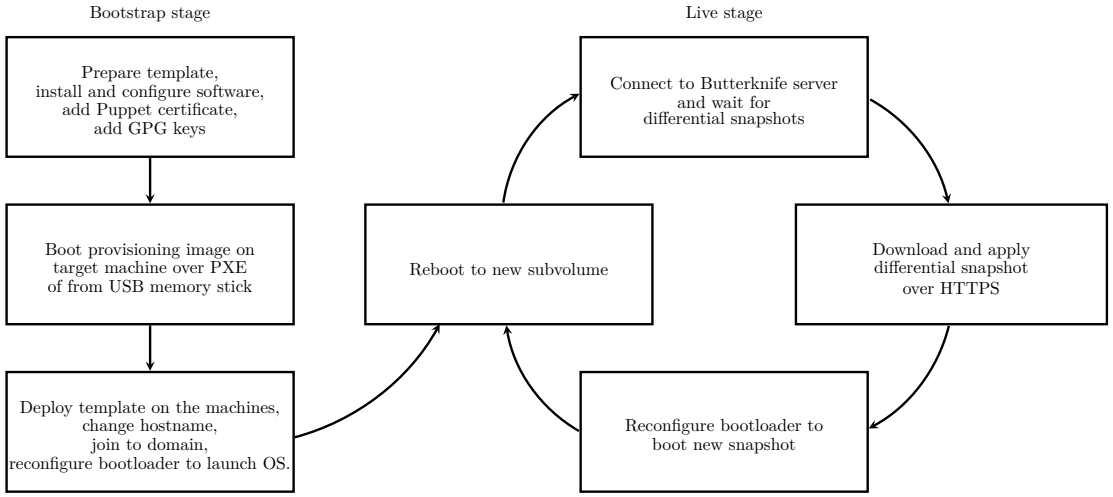

Figure 3.2. Butterknife deployment workflow.

Butterknife consists of four major components: Template helpers, command-line utility, Buildroot based provisioning image and services for applying incremental snapshots online.

3.2 Template helpers

LXC containers are used to bootstrap the template for provisioning. Creating container with Btrfs backing store on top of a Btrfs filesystem places the container in an isolated subvolume which makes it easy to snapshot the container. Within the container *puppet apply* and similar methods can be used to take advantage of already existing configuration management know-how. Otherwise traditional manual labour can be employed to set up the template: installing packages, tweaking configuration files etc. Post-deploy scripts are placed under `/etc/butterknife/post-deploy.d` – Butterknife helpers to provide persistent hostname, domain join, Puppet certificates and much more.

During the release phase the LXC container is stopped, pre-release scripts placed under `/etc/butterknife/pre-deploy.d` are executed to clean up package cache, Fontconfig caches, locale cache and other temporary files. Then a read-only Btrfs snapshot is generated from the container root filesystem. At this point new snapshot becomes available via running instance of `butterknife serve`

3.3 Command-line utility

The command-line utility is mainly targeted for advanced usecases: pushing-pulling snapshots via Secure Shell (SSH), pulling snapshots via HTTP(S) or multicast, serving snapshots via HTTP or multicast. The utility also provides interfacing with LXC to prepare the templates and support for `systemd-nspawn` is planned in future.

The Python programming is done in object-oriented manner, thus adding additional transport methods is trivial. Falcon HTTP API framework [59] from Rackspace was used to build the HTTP API portion of the Butterknife to serve snapshots over HTTP. The HTTP API serves information about templates available in the server's `/var/butterknife/pool`. The API exposes several methods for iterating over templates and subvolumes present in the server's Btrfs pool.

The *name* of a template follows naming scheme of DBus objects incorporating fully qualified domain name in reverse and the identifier of the object. The *version* refers to the snapshot of the template. The *arch* refers to target architecture which is normalized to *x86* for 32-bit and *x86_64* for 64-bit Intel x86 machines. For example the Btrfs subvolume stream URL for snap42 of the 32-bit `EduWorkstation that originates from butterknife.koodur.com` would be `/api/template/com.koodur.butterknife.EduWorkstation/arch/x86/version/snap42/stream`.

Note that the stream URL also accepts parent argument, so incremental snapshot can be received simply by appending *?parent=snap41*. The unicast snapshot transfer topology is shown in Figure 3.3. Note that unicast suffers obvious scalability issue, the uplink of the server is eventually congested and throughput per node decreases with every additional node.

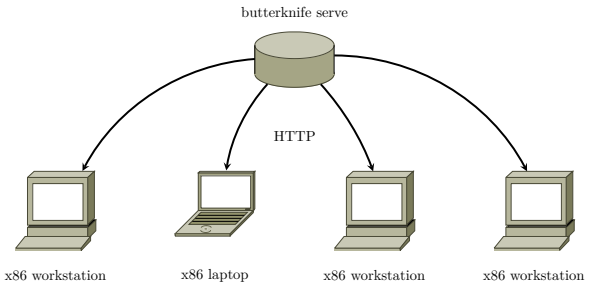

Figure 3.3. Deployment over HTTP.

Multicast is used to resolve the scalability issue of the initial provisioning stage. Snapshot transfer topology with off-site server is shown in Figure 3.4. In this case the snapshot is transferred over HTTP from the server by one of the participating nodes. That node proxies the stream to local LAN segment using multicast. All the other nodes are receiving over multicast

Alternatively snapshots can be pulled either over HTTP or SSH from the server to a local machine and served from there over multicast as shown in Figure 3.5.

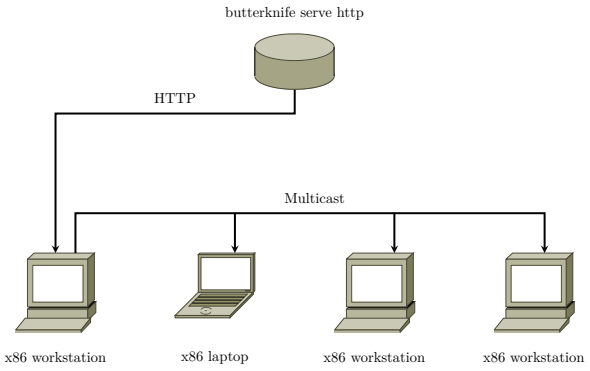

Figure 3.4. Deployment over HTTP and multicast.

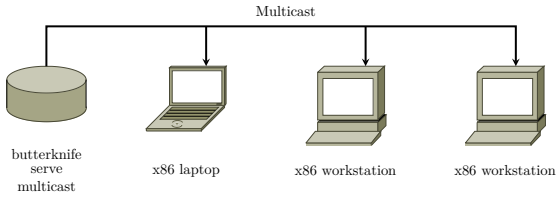

Figure 3.5. Deployment in local network segment over multicast.

3.4 Provisioning image

The provisioning image was designed to bootstrap a machine. Debian, Ubuntu, Gentoo and Buildroot were evaluated as provisioning platforms. With Debian and Ubuntu the resulting PXE bootable image would have exceeded 100MB, as the image is uncompressed to RAM, booting on machines with less than 1GB of RAM is troublesome. As of February of 2015 the CoreOS image suffers similar issue - *vmlinuz* [1] and *initrd* [2] files required to boot over PXE are correspondingly 24MB and 117MB. With Gentoo significant tweaking was required, because Gentoo is mainly targeted for power users. Buildroot an embedded Linux system build system was eventually used to generate an (<15MB) all-in-one PXE-bootable image as well as ISO image which can be dumped on a USB memory stick [60].

Using Python to build pseudo-graphic menu-driven user interface was evaluated and deemed not necessary for the goal as Python runtime and dependant libraries add about 10MB to the resulting image. Additionally `parted` Python bindings were unavailable in Buildroot package selection. Instead `dialog` in conjunction with `curl`, `jq` and others were used to build the user-interface and shell was used to program the user-interface logic. Resizing of NTFS filesystems is provided by `ntrfsresize`

[1] `http://stable.release.core-os.net/amd64-usr/current/coreos_production_pxe.vmlinuz`

[2] `http://stable.release.core-os.net/amd64-usr/current/coreos_production_pxe_image.cpio.gz`

utility which is part of `ntfs-3g` package, this eases deployment of dual-boot machines. Complete multicast is supported via consistent snapshot naming scheme and `udpcast -receive` and `udpcast-sender` utilities which are part of `udpcast` package.

The security model for the initial deployment phase could be improved as only method of verification of the source is the certificate authority chain verified by *curl* during the Btrfs snapshot retrieval. For multicast transfers there is no security enforced by software, it is assumed that multicast is used only in protected LAN segments.

The used partitioning scheme is described in Figure 3.6 is inspired by Ubuntu and Lennart Poettering's article. [61] The unallocated space of block device is used create single Btrfs filesystem which is mounted at `/var/butterknife/pool` making it possible to easily iterate over templates and root filesystem instances present in the machine. Whenever a Btrfs (differential) snapshot stream is received the incoming subvolume is placed under `/var/butterknife/pool`. Before transfer remnants of previously interrupted transfers are cleaned up by simply iterating over directories beginning with `@template:` and attempting to create a file inside the directory. If the file creation succeeds, the subvolume is deleted. Note that prior successful exit `btrfs receive` sets the subvolume read-only, thus file creation should fail.

Once the template subvolume is successfully received a read-write clone is made with `@root:` prefix, this signifies an *instance* of a root filesystem. The instance subvolume is mounted at `/mnt/target` and several mountpoints such as `/dev/`, `/proc/`, `/sys/` are bound to `/mnt/target`.

Finally `chroot` is issued to enter the instance to run `butterknife-postdeploy` which executes post-deploy scripts which are part of the template. Most usually the Btrfs pool is mounted at `/var/butterknife/pool` of the instance and `@persistent` subvolume is created in the pool. The persistent subvolume is then mounted at `/var/butterknife/persistent`. The persistent subvolume is used to retain hostname, domain join information (Kerberos keytab, Samba secrets), Puppet keys and certificates etc. Final aspect of the post-deploy scripts is the bootloader installation. As the root filesystem is essentially swapped out, `grub-install` is required to update references to the newly created OS instance.

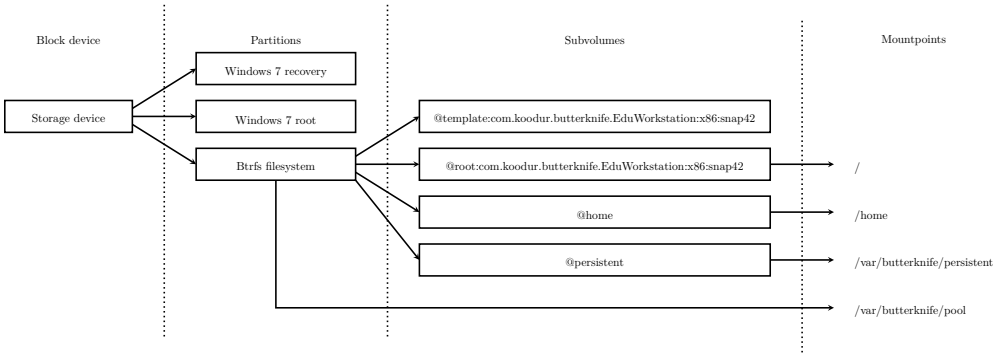

Figure 3.6. Pooled partitioning.

3.5 BitTorrent integration

Multicast is designed for usecases where simultaneous control over multiple machines is possible. The differential snapshots are primarily downloaded via HTTP(S) because coordinating multicast transfers with roaming laptops is a complex scenario. Optionally push/pull can be performed over SSH making Butterknife already now very flexible tool.

BitTorrent is a peer-to-peer (P2P) file sharing protocol designed by Bram Cohen [62]. BitTorrent is designed to facilitate file transfers among multiple peers across unreliable networks. It has potential to offload content transfer to nodes participating in the network as shown in Figure 3.7. In this section two viable approaches are discussed.

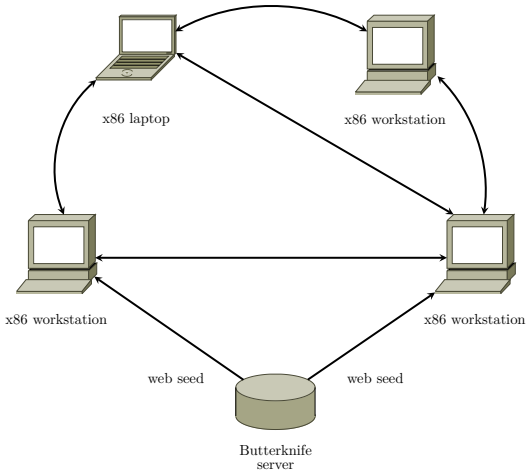

Figure 3.7. Possible load distribution scenario using BitTorrent.

As getting consistent output from `btrfs send` is tricky it makes sense to store the snapshot bitstreams as files and use BitTorrent to redistribute them. This way `received_uuid` can be kept in sync with the origin. Similarly each incremental snapshot can be stored on the disk and distributed as torrents. As full snapshots are large and consume a lot of disk space it makes sense for example keep every 100th full snapshot and apply incremental snapshots one by one to get to a desired snapshot. The main advantage of such approach is the overhead: in addition to Btrfs filesystem which already contains all the necessary snapshots, full and incremental snapshot bitstreams are stored as separate files. Additionally the step-by-step incremental snapshot approach defeats the whole purpose of using filesystem such as Btrfs.

Second approach dives into the BitTorrent specifics. A .torrent file contains metadata about the content in question and a tracker URL – that is the service which is used to discover other peers in the pool of participating nodes also known as swarm. BitTorrent splits files into pieces and SHA-1 hash is calculated per piece. BitTorrent protocol does not specify minimal piece length [63], but for example

libtorrent imposes restriction of having piece length a multiple of 16KB [64]. It is recommended to keep BitTorrent file size below 100kB, which means the piece size is correlated to content size. [65].

For multi-file torrents the files in the directory tree are handled as a continuous stream of data of the concatenated files as shown in Figure 3.8, thus changing size of a file that happens to be placed in the beginning of a torrent file results in completely different checksums for the whole torrent. Such approach is reasonable for rarely changing data, but for current usecase causes significant overhead.

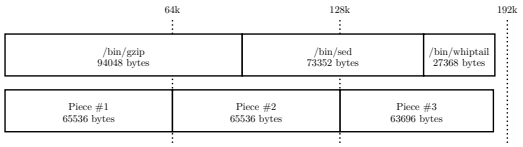

Figure 3.8. Multi-file torrent handles directory tree as a continuous stream of data.

BitComet has implemented *Align File to Piece Boundary* function [66], which adds a padding file if necessary to align files to piece boundary as shown in Figure 3.9. This way pieces which contain identical files retain same piece checksums, however small files add significant overhead due to padding files. Compression of pieces sent on the wire has been proposed [67]. This could resolve the overhead issue introduced due to padding files. The problem of piece alignment could also be addressed simply by introducing new file mode.

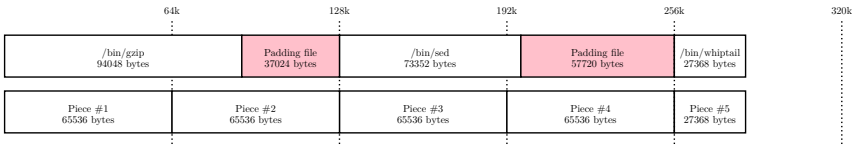

Figure 3.9. Align File to Piece Boundary.

Btrfs uses CRC-32C for checksumming and support for additional checksum algorithms, namely SHA-1 is planned [68]. Current on-disk format supports up to 256-bit hash checksum per metadata block and arbitrary count of hashes for per-block checksums. [69]. As of November 2013 Btrfs defaults to 16kB or page size whichever is larger. On most Linux workstation page size is set to 4kB, thus 16kB block size takes precedence [70].

As online deduplication is in works for Btrfs it makes sense to combine the two, implementing additional system call for Btrfs in order to perform block lookup by checksum is trivial task. This could make it possible to implement high-performance BitTorrent implementation which takes advantage of Btrfs metadata.

Aligned checksumming would permit sharing platform-independent (images; fontconfig cache; dconf database; LaTeX packages; Bash, Python, Ruby, Perl, Lua, Java source and bytecode) file chunks between machines of different architecture

(amd64, i386, armel, armhf) from arbitrary snapshots.

Generating .torrent corresponding to a root filesystem has other issues as well. Piece size of 16kB results in a torrent file exceeding 10MB for Ubuntu root filesystem and that did not even include padding files required to align piece boundary to file beginning. Most BitTorrent client implementations fail to handle .torrent files of such size resulting in out of memory errors or freezes. Also BitTorrent currently does not handle symlinks, POSIX filesystem permissions and access control lists [71].

This leads us to believe that these properties be transferred using additional manifest file generated by eg `fssum`, making use of .torrent file redundant. In fact Google Summer of Code 2015 project [72] was submitted by Carnegie Mellon University graduate student Harshad Shirwadkar to implement content based storage in Btrfs [73].

Bram Cohen, the BitTorrent creator filed patent for live streaming version of BitTorrent [74] which promises latency of few seconds opposed to previously available approaches which exceeded latency of several magnitudes higher. The technology will be available free of charge for users as well as content distributors. Patent will be used to enforce the quality of the available software so for example client applications of bad quality would not degrade the performance of the technology. BitTorrent Live could be used to synchronize several clients to apply differential snapshot from same parents to same target snapshots and use BitTorrent Live simply as transport protocol.

Chapter 4

Implementation issues

4.1 Btrfs receive parent subvolume lookup

As of kernel 3.17 there is no consistent way of transferring differential snapshots in a distributed Git-like fashion due to way `btrfs receive` locates the parent subvolume. Whenever `btrfs send` is issued, the *uuid* of origin subvolume is bundled with the bitstream, that becomes the *received_uuid* on the receiving endpoint and new *uuid* is assigned for the created subvolume. For incremental snapshots `btrfs send` also bundles the `uuid` of the parent subvolume, now `btrfs receive` attempts to locate the parent subvolume by *received_uuid* of local subvolumes.

This effectively restricts the workflow to one direction as shown in Figure 4.1. In other words if a full snapshot is transferred from machine A to B and B to C, then it is impossible to apply incremental snapshots from A to C, because `btrfs receive` is unable to locate the snapshot to be used as parent. This prevents running a downstream Butterknife server and even more importantly it makes tricky restoring snapshot in scenarios where Btrfs is used for backing up.

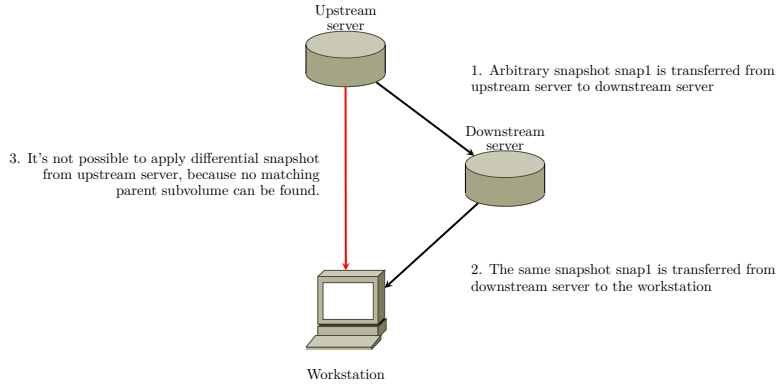

Figure 4.1. `received_uuid` inconsistency.

Arne Jansen, previously a Btrfs developer maintains far-progs repository [75],

which contains tools for manipulating Btrfs snapshot streams, this format is also known as FAR (Filesystem ARchive). The `btrfs-receive` from `far-progs` features two extra flags: `-p` which disables automatic parent searching and allows user to specify which subvolume to use as a parent and `-d` which allows specifying a custom subvolume name. Disabling parent search makes it possible to detach Btrfs' UUID mechanism and customize the differential snapshot logic. In this case care must be taken of snapshot consistency, applying differential snapshot on incorrect parent subvolume may cause data corruption or end up with a crash of `btrfs receive`. The changes were isolated by the author and submitted to `btrfs-progs` upstream [1].

Note that current `btrfs receive` approach does not provide any verification of the resulting snapshot and the unique identifiers are simply used to look up the parent subvolume. Thus disabling automatic parent search and using the Butterknife provided logic is not necessarily inferior method.

In fact far-progs repository contains `fssum` utility which can be used to checksum a directory tree. It checks both – file contents and attributes, including permissions and modification times. This information could be for example signed by GPG in the developer machine and published via `butterknife serve` to downstream machines for verification purposes.

4.2 Btrfs receive confinement

As said, Btrfs is an evolving filesystem and there are corner-cases where Btrfs fails security-wise. For example issuing `btrfs subvol list` within a `chroot` exposes subvolumes outside the new root filesystem.

For Butterknife usecase other security aspects have to be considered. As of *btrfs-progs* version 3.19 the `btrfs send` emits simply a stream of encoded opcodes which are intended to be replayed by `btrfs receive`: create subvolume, snapshot parent subvolume, create file, *mkdir*, *mknod*, *mkfifo*, symlink, *link*, *unlink*, *rename*, *rmdir*, open file, close file, write to file, set/remove extended attributes, truncate file, *chmod*, *chown* [2].

In case of full snapshot system call is issued to create empty subvolume with specified name and in case of differential snapshot an system call is issued to locate parent subvolume by UUID and clone it. After the initial opcode traditional filesystem operations are carried out on the newly created/cloned subvolume.

As can be seen in the `unlink` opcode implementation on line 456 of cmds-receive.c [3], the `path_cat` function is used to concatenate path of the subvolume and target filename. Investigating further reveals that no sanitization is performed on the

[1] [PATCH] btrfs-progs: receive explicit parent support `http://www.mail-archive.com/linux-btrfs@vger.kernel.org/msg43329.html`

[2] `http://git.kernel.org/cgit/linux/kernel/git/kdave/btrfs-progs.git/tree/cmds-receive.c?h=v3.19.x#n792`

[3] `http://git.kernel.org/cgit/linux/kernel/git/kdave/btrfs-progs.git/tree/cmds-receive.c?h=v3.19.x#n456`

path [4], leaving receiving end vulnerable to path traversal attacks. Carefully crafted Btrfs stream could for example delete files of the running operating system rendering the whole system in unusable state, or even worse substitute for example Puppet configuration and certificate authority file in the Butterknife persistent subvolume making it possible for a third party to gain access over the machine.

Several `btrfs-progs` patches were submitted to improve the security of `btrfs receive`. Patch to enforce `chroot` before parsing the Btrfs stream was accepted upstream [5]. This patch however confines the `btrfs receive` process only to the parent directory of the newly created subvolume, making it possible to attack other subvolumes contained in the same directory. The confinement could be improved even more, for example by parsing the first opcode, performing either subvolume creation or parent subvolume cloning, then issuing `chroot` to the subvolume directory and finally replaying the remaining opcodes. This would, of course make it troublesome to receive multiple snapshots in the same stream, as it is possible via `-e` flag of `btrfs send`.

4.3 Bootloader

GRand Unified Bootloader (GRUB) is the main bootloader used by Linux-based operating systems on x86 and PowerPC based machines. Earlier versions of GRUB supported multi-stage booting process, which meant that GRUB stage1 binary was embedded in the Master Boot Record (MBR) which loaded stage1.5 embedded 32256 byte area between the MBR and first partition. The stage1.5 contained filesystem drivers which could address the filesystem contents and boot stage2 from the `/boot/grub` of the Linux filesystem. [76]

GRUB2 added support for booting from Btrfs root filesystem in various configurations hence in order to boot from Btrfs a GRUB2 installation is required and older versions of GRUB are not supported. GRUB2 also now supports booting from a particular subvolume, making it possible to place several root filesystems in same Btrfs pool. [77]

GRUB2 dropped support for multi-stage booting process, instead it is recommends that the first partition starts at megabyte boundary, leaving more than 500kB room between MBR and first partition which is well enough to accommodate feature-rich bootloader such as GRUB with Btrfs support. Windows XP partitioning however does not leave enough room to accommodate GRUB2 with Btrfs support, resulting in obscure errors. As a workaround *ms-sys* [78] utility can be used to restore Windows XP master boot record and install GRUB in the volume instead. Subsequently *fdisk* can be used to set that volume active.

[4] `http://git.kernel.org/cgit/linux/kernel/git/kdave/btrfs-progs.git/tree/send-utils.c?h=v3.19.x#n711`

[5] [PATCH] btrfs-progs: enforce chroot for btrfs receive `https://www.mail-archive.com/linux-btrfs@vger.kernel.org/msg43019.html`

GUID Partition Table (GPT) was introduced to overcome limitations of Master Boot Record, such as maximum addressable space of 2TB [79]. Unified Extensible Firmware Interface (UEFI) was originally introduced by Intel and it mandates the use of GPT [80]. Most modern PC-s and Macs have substituted legacy PC BIOS with UEFI. In order to boot GRUB2 on UEFI the GRUB binary has to be either placed in the FAT32-formatted EFI partition with partition type *ef00* and proper EFI firmware entry has to be created using *efibootmgr* [81]. If the machine does not support true UEFI or there is another reason to use legacy boot methods on 2TB+ disks there is an option to make use of BIOS boot partition with partition type *ef02* which can be used to embed GRUB2 in case of a GPT partition table.

Current implementation of Butterknife relies on legacy MBR partitioning scheme, but the groundwork has been laid to support GPT and hybrid partitioning schemes.

Chapter 5

Experimental Evaluation

In order to show that our basic approach in Butterknife is practical and and extendible for large-scale applications such as unrolling updates for workstations, netbooks, smartphones, automotive applications or other embedded devices an experimental evaluation was carried out on the prototype.

5.1 Performance

To evaluate performance of Butterknife an arbitrary production snapshot was chosen. The `@template:com.koodur.butterknife.EduWorkstation:x86:snap83` snapshot was based on 32-bit Ubuntu 14.04 and it included MATE 1.8.2 desktop environment, Mozilla Firefox 37 web browser, LibreOffice 4.4.2 office suite, VLC 2.1.6 multimedia player, Skype 4.3.0 voice over IP solution, Inkscape 0.48.4 scalable vector graphics editor, Gimp 2.8.10 photo editor and many tweaks to make the system usable in Estonian schools.

The throughput of the server-client channel depends on several factors: raw snapshot bitstream size, network bandwidth; compression algorithm; parallelization of compression; encryption; storage medium.

The disk usage of the subvolume was approximately 6.20GiB due to fragmentation and holes in sparse files. Issuing `btrfs send` on the subvolume results in approximately 5.75GiB bitstream. Compressing the bitstream yields different results depending on the compression algorithm as shown in Figure 5.1 ranging from 3.21GB using *lz4* to 1.87GB using *xz*.

Note that streaming compression is necessary for scenarios where differential snapshots are not stored as static files or where the full snapshots are directly served from `btrfs send` which is also the default case for Butterknife. Direct streaming with `btrfs send` makes it possible to take full advantage of the copy-on-write filesystem and snapshots while keeping disk usage on the server side minimal.

Various compression tools were benchmarked with the raw Btrfs stream stored in RAM. The time shown in Figure 5.2 was measured on Intel(R) Core(TM) i7-4770R quad-core processor clocked at 3.20GHz with 2x 8GB DDR3 memory modules run-

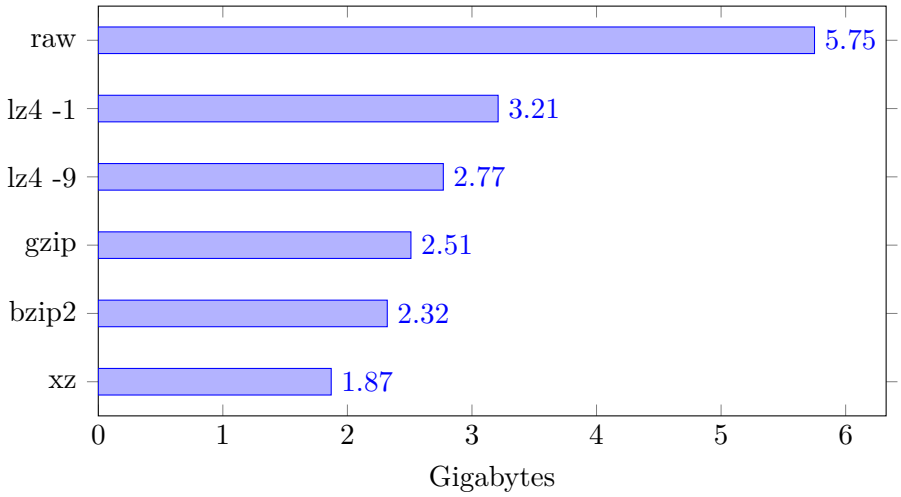

Figure 5.1. Compressed Btrfs bitstream disk usage.

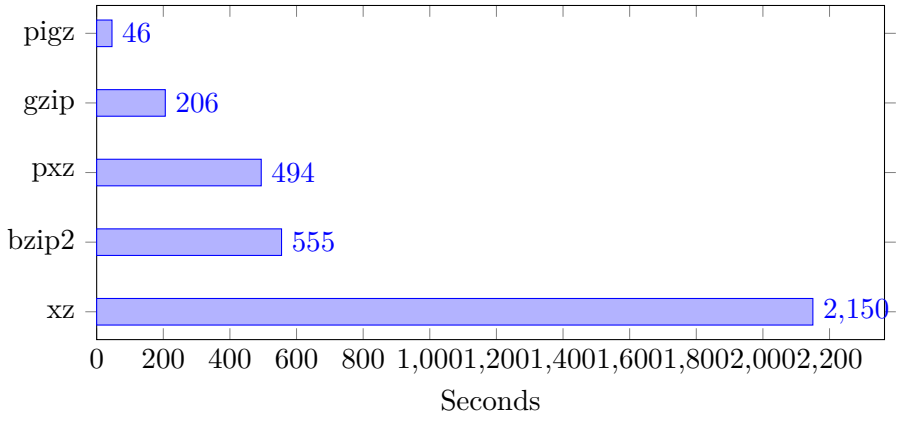

Figure 5.2. Btrfs stream compression time on Intel i7-4770R.

ning Ubuntu 14.04 LTS and 3.16.0 kernel. The source and destination directories were mounted as `tmpfs` on the same machine to exclude the storage and network effects on the test. In addition to single-threaded `gzip` the parallel version of `gzip` algorithm, `pigz` is included in the results as well as `pxz`, the parallel version of `xz` . Similar tests were performed on QNAP TS-451 NAS-box equipped with Intel Celeron J1800 dual-core processor and 4GB DDR3 memory module.

As root filesystem contains numerous small files significant slowdown was imminent if spinning disk was used on the either side. An average throughput of 37MB/s was observed while running `btrfs send` from or `btrfs receive` to Western Digital WD40EFRX without using encryption or compression over plain HTTP. Using arbitrary SSD-s on both ends averaged around 100MB/s due to gigabit eth-

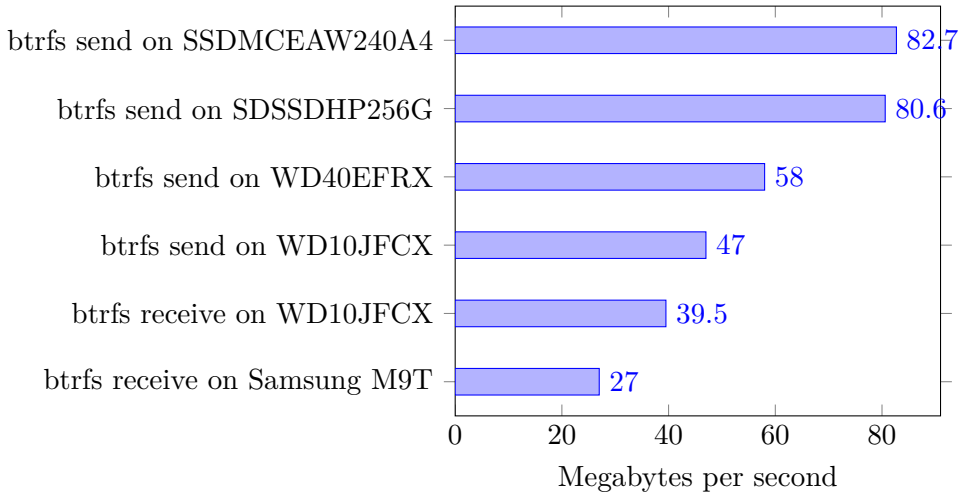

Figure 5.3. Average throughput for various storage devices.

ernet used for benchmarking. SSD-s however face other issues, the block remapping mechanism used by SSD-s for wear levelling gradually degrades the write performance of SSD if *fstrim* [82] is not regularly used to release unused blocks.

The benchmark results for Intel 240G mSATA SSD (SSDMCEAW240A4), Sandisk 256G SATA SSD (SDSSDHP256G), Western Digital Red 4TB SATA 3.5" HDD (WD40EFRX), Western Digital Red 1TB SATA 2.5" HDD (WD10JFCX) and Samsung M9T 2TB SATA 2.5" HDD are shown in Figure 5.3. The `btrfs receive` is generally slightly slower compared to `btrfs send` for the same device, presumably due to read-ahead caching mechanism of the block device.

The empiric observations carried out with the snap83 of the root filesystem as shown in Figure 5.4 pointed out the bottlenecks that previously went unnoticed. The most important conclusion that can be drawn from the results is that enabling compression may actually hamper the deployment speed. Single-threaded `gzip` might make sense for 100MBps infrastructure, but going beyond gigabit threshold the law of diminishing returns can be observed. Additionally the hard disks with spinning platters impose limitations caused by prolonged seek times. Even with solid-state disks effects of normal filesystem fragmentation can be observed. The compression aspect is notable because Estonian Educational Network has scheduled infrastructure upgrades for 2016 and their plan is to supply gigabit link to every educational institution of Estonia.

5.2 Integrity

Btrfs is relatively new filesystem and even though on-disk format of Btrfs is not expected to change any more the kernel drivers and userspace utilities are still being actively developed. Due to this there are certain corner-cases where Btrfs may

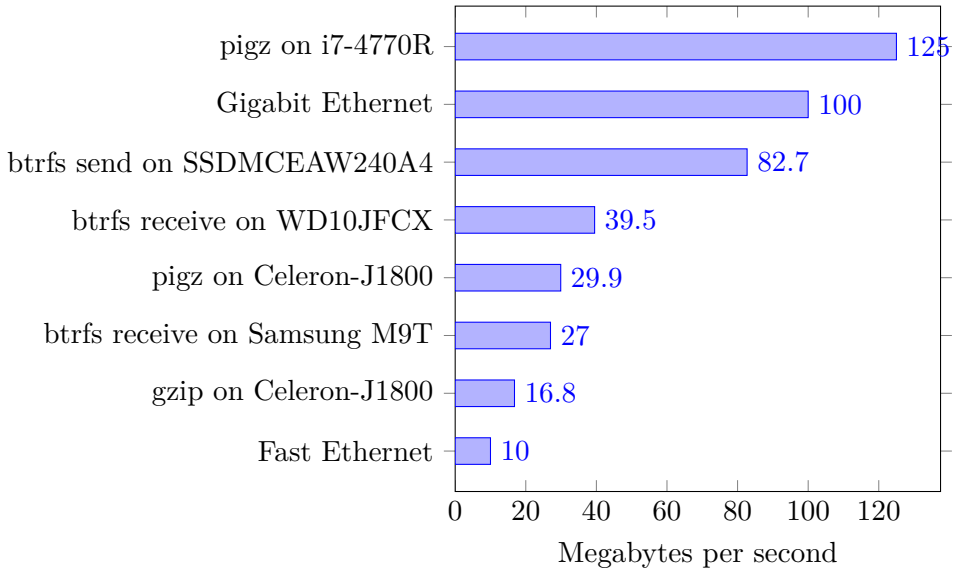

Figure 5.4. Identified bottlenecks.

fail horribly, for example 3.14 and earlier kernel had often issues with filesystem corruption and data loss [83], especially when filesystem was running out of free space. The author has not faced any serious issues with 3.16 kernel, but for example btrfs-progs 3.16 have bugs which prevent it's use in case of nested subvolumes. Ubuntu also has upgraded from 3.14 to 3.16 in Ubuntu 14.04, their last Long Term Support release. Linux 3.18 has also proven to be reliable when in comes to Btrfs, but for example Linux 3.19 faces a deadlock during an attempt to mount dirty Btrfs filesystem [84].

5.3 Usability

Butterknife omits the overhead and complexity associated with traditional ISO remastering systems such as *remastersys* [52] making it possible for virtually anyone to roll a customized distribution for particular purpose. Butterknife also makes it easier to bundle proprietary components into the template, thus setting up APT repository to distribute packages for the targets is not necessary. Even though online updating is not implemented yet, upgrading systems via provisioning tool is already convenient. The persistent subvolumes make it trivial to switch from 32-bit (x86) root filesystem to 64-bit (x86-64) and vice versa.

Butterknife project users, fellow devops in this case were positively surprised by the streamlined deployment process of Butterknife. Butterknife service users, local IT-support in this case were generally satisfied with the delivery. Reinstallation of classroom or conversion to dual-boot, was not any more necessarily a task that had to be strictly scheduled for school holidays. Butterknife-based upgrade from

Ubuntu 12.04 to Ubuntu 14.04 was performed within first half of a day at Tallinn Mahtra Primary School. Butterknife was also successfully used at Tallinna Humanitaargümnaasium and Tallinna 32. Keskkool to convert Windows 7 machines to dual-boot.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The Butterknife addresses current issues of Linux-based operating system deployment: time-consumption, complexity, reliability and customizability. The Butterknife implementation as of May 2015 is ready for preparing Linux-based operating system templates and deployment of the templates. Butterknife was used on 2nd of May to deploy machines as part of Vabavaratalgud [85] effort to convert unsupported Windows XP machines to modern open-source operating system. Local Butterknife instance was used to serve templates in local LAN segment drastically reducing the deployment time of the template.

The Butterknife implementation satisfies, exceeds to be precise, the requirements of educational institutions. It significantly decreases local IT-support work, by reducing workstation bootstrap down to 15 minutes even for a regular sized classroom. With the addition of push/pull capabilities Butterknife is also becoming attractive tool for devops who need to move around Linux containers.

Source code of the solution was published at GitHub. [1] The instructions for setting up similar infrastructure are provided at GitHub and are constantly being improved.

6.2 Adding verification support

Currently there is no method for verifying the integrity of received snapshot – the transport channel security (TLS, SSH) is the only method against man-in-the-middle attacks. There are plans to make use of *fssum* to generate manifest of the snapshot directory tree and sign the file with GPG on the developer machine. Estonian ID-card or any other hardware crypto token can be used to add extra layer of security.

An extra command *verify* for *butterknife* utility will be added to generate manifest on the receiving end and to compare it's contents agains the signed manifest.

[1] GitHub: Butterknife provisioning suite `https://github.com/laurivosandi/butterknife/`

The manifest comparison time can be reduced by parsing verbose output of `btrfs receive` to determine modified files in case of differential snapshots. Additionally Butterknife has to be modified to incorporate signature querying method over the transport layer.

6.3 Adding online snapshot retrieval

Currently the machine has to be temporarily booted to provisioning image in order to deploy incremental snapshots. The Btrfs allows retrieving snapshots while the filesystem is mounted and the templates can be pushed/pulled already now, however online deployment of the templates is not yet implemented.

Additionally facilities to notify user about availability of new templates, confirming an upgrade and scheduling a reboot to new deployment have to be implemented. There are plans to make use of DBus and Python to build a service for querying remote Butterknife server about availability of upgrades, downloading the updates and deploying the updates – in fact this could be implemented as part of Butterknife command-line utility.

Another DBus service written in Python can be used to implement tray icon service making use of graphical user interface libraries such as GTK or Qt. The tray applet can then be used to present computer user information about the updates and telling the computer user to reboot into the new version of root filesystem.

Note that for certain usecases, where there is organization-wide enforced policy to unroll security updates as soon as possible, the tray applet is simply used to notify user about enforced reboot and the user might not be given any opportunity to schedule an update.

6.4 Adding systemd-nspawn support

Currently LXC is the only supported template preparation method. Debian 8 codenamed *jessie* was released in the April of 2015 with a new default init system *systemd* [86]. Ubuntu 15.04 released few days earlier also ships with *systemd* by default [87]. As *systemd-nspawn* and *machinectl* are bundled with *systemd*, it makes sense to take advantage of the ecosystem and provide even better integration with Butterknife – For example implementing *butterknife enter* command for launching and entering an *systemd-nspawn* container on-demand becomes trivial.

6.5 Improving scalability

Currently HTTP(S), SSH and multicast are available for transport and even though BitTorrent was discussed in the Section 3 – Butterknife Design and Architecture, no implementation attempts were made. To further improve Butterknife scalability integration of BitTorrent, BitTorrent Live or similar technology would be necessary. Identifying production-ready approach requires more experimentation though.

6.6 Alternative filesystem layouts

Lennart Poettering, a controversial Free Software developer has outlined a method of building Linux based systems using Btrfs snapshots [61]. The new layout splits high level components of software (operating system, desktop environment, frameworks, office suit) into separate subvolumes which can be upgraded independently. This requires significant effort from operating system distributors and software suite vendors, but promises significant save of effort on testing software on Linux based systems. Most notably a snapshot naming scheme is proposed in the article, which would permit mixing operating system files with different sets of libraries, frameworks and applications. Poettering's naming scheme was used to derive the Butterknife naming scheme.

Bibliography

[1] "Ubuntu pre-installed." [Online]. Available: https://help.ubuntu.com/community/UbuntuPre-installed

[2] L. Torvalds, "Linux: a portable operating system," Master's thesis, University of Helsinki, January 1997. [Online]. Available: http://mirror.linux.org.au/pub/linux/kernel/people/torvalds/thesis/torvalds97.pdf

[3] R. M. Stallman and J. Gay, "Free software, free society: Selected essays of richard m. stallman," 2002. [Online]. Available: https://www.gnu.org/philosophy/fsfs/rms-essays.pdf

[4] J. Ong, "Android reached record 85% smartphone market share in q2 2014," Jul. 2014. [Online]. Available: http://thenextweb.com/google/2014/07/31/android-reached-record-85-smartphone-market-share-q2-2014-report/

[5] R. Myslewski, "Windows hits the skids, mac os x on the rise," Mar. 2014. [Online]. Available: http://www.theregister.co.uk/2014/03/15/windows_desktop_and_laptop_market_share_dips_below_90_per_cent/

[6] "Puppet labs: It automation software for system administrators." [Online]. Available: http://puppetlabs.com/

[7] A. Tsalolikhin, "The state of open source system automation," *Linux Magazine*, Aug. 2010. [Online]. Available: http://www.linux-mag.com/id/7841/

[8] "Introduction to salt." [Online]. Available: http://docs.saltstack.com/en/latest/topics/

[9] "The raet transport." [Online]. Available: http://docs.saltstack.com/en/latest/topics/transports/raet/index.html

[10] "Fai - fully automatic installation." [Online]. Available: http://fai-project.org/

[11] "Clonezilla." [Online]. Available: http://clonezilla.org/

[12] "Partclone." [Online]. Available: http://partclone.org/

[13] "Symantec ghost solution suite." [Online]. Available: http://www.symantec.com/ghost-solution-suite/

[14] "Acronis true image." [Online]. Available: http://www.acronis.com/en-eu/personal/pc-backup/

[15] "Fsarchiver: Filesystem archiver for linux." [Online]. Available: http://www.fsarchiver.org/

[16] M. Riondato, "Freebsd handbook: Chapter 15. jails." [Online]. Available: https://www.freebsd.org/doc/en/books/handbook/jails.html

[17] "Linux containers." [Online]. Available: https://linuxcontainers.org/

[18] J. Edge, "Creating containers with systemd-nspawn," *LWN.net*, Nov. 2013. [Online]. Available: https://lwn.net/Articles/572957/

[19] N. McAllister, "Coreos's docker-rival rocket: We drill into new container contender." [Online]. Available: http://www.theregister.co.uk/2014/12/03/coreos_rocket_deep_dive/

[20] "Coreos." [Online]. Available: https://coreos.com/

[21] "Snappy ubuntu." [Online]. Available: http://developer.ubuntu.com/en/snappy/

[22] N. Brown, "Overlay filesystem," Oct. 2014. [Online]. Available: https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/Documentation/filesystems/overlayfs.txt

[23] J. Corbet, "Fedora and lvm," Oct. 2012. [Online]. Available: http://lwn.net/Articles/522073/

[24] O. Rodeh, J. Bacik, and C. Mason, "Btrfs: The linux b-tree filesystem," 2012. [Online]. Available: http://www.cse.iitb.ac.in/~johncf/papers/btrfs.pdf

[25] C. Mason, "The btrfs filesystem," Sep. 2007. [Online]. Available: https://oss.oracle.com/projects/btrfs/dist/documentation/btrfs-ukuug.pdf

[26] "Btrfs wiki: Production users." [Online]. Available: https://btrfs.wiki.kernel.org/index.php?title=Production_Users&oldid=29151

[27] C. Mason, "linux-btrfs mailinglist: Leaving fusion-io," Nov. 2013. [Online]. Available: http://article.gmane.org/gmane.comp.file-systems.btrfs/30420

[28] ——, "Btrfs for 3.19-rc," Dec. 2014. [Online]. Available: http://lkml.iu.edu/hypermail/linux/kernel/1412.1/03583.html

[29] "Debian wiki: Btrfs," Sep. 2013. [Online]. Available: https://wiki.debian.org/Btrfs

[30] "apt-btrfs-snapshot." [Online]. Available: https://launchpad.net/apt-btrfs-snapshot

[31] “yum-fs-snapshot.” [Online]. Available: http://man7.org/linux/man-pages/man1/yum-fs-snapshot.1.html

[32] M. Chiappetta, “Turn old pc hardware into a home server with freenas,” Jun. 2014. [Online]. Available: http://www.pcworld.com/article/2243748/turn-old-pc-hardware-into-a-killer-home-server-with-freenas.html

[33] J. Corbet, “Rockstor - a btrfs-based nas distribution,” Jan. 2015. [Online]. Available: http://lwn.net/Articles/629908/

[34] ——, “Openmediavault: a distribution for nas boxes,” Jan. 2015. [Online]. Available: http://lwn.net/Articles/628284/

[35] “Free software foundation: More background about the cisco case.” [Online]. Available: http://www.fsf.org/licensing/2008-12-cisco-complaint

[36] “www.dd-wrt.com - unleash your router.” [Online]. Available: http://www.dd-wrt.com/

[37] “polarcloud.com - tomato firmware.” [Online]. Available: http://www.polarcloud.com/tomato

[38] “Openwrt - wireless freedom.” [Online]. Available: https://openwrt.org/

[39] “Openwrt: Table of hardware.” [Online]. Available: http://wiki.openwrt.org/toh/

[40] “Openwrt wiki: The openwrt flash layout.” [Online]. Available: http://wiki.openwrt.org/doc/techref/flash.layout

[41] “squashfs - a compressed filesystem for linux.” [Online]. Available: http://sourceforge.net/projects/squashfs/

[42] G. Ben-Yossef, “Building murphy-compatible embedded linux systems,” Nov. 2014. [Online]. Available: https://blog.feabhas.com/wp-content/uploads/2014/11/Writing-a-Linux-Update-Mechanism.pdf

[43] A. Vandecapelle, “Safe upgrade of embedded systems,” 2012. [Online]. Available: https://archive.fosdem.org/2012/schedule/event/699/130_Presentation_Safe-Upgrade.pdf

[44] H. Raja, “What is clockworkmod recovery and how to use it on android,” Mar. 2011. [Online]. Available: http://www.addictivetips.com/mobile/what-is-clockworkmod-recovery-and-how-to-use-it-on-android-complete-guide/

[45] “Teamwin - twrp.” [Online]. Available: http://teamw.in/project/twrp2

[46] “Cyanogenmod wiki: Fastboot intro.” [Online]. Available: http://wiki.cyanogenmod.org/w/Doc:_fastboot_intro

[47] O. Foundation, "Welcome to glance's documentation," February 2015. [Online]. Available: http://docs.openstack.org/developer/glance/

[48] A. Messerli, "Launchpad blueprints: Glance bittorrent delivery," Aug. 2011. [Online]. Available: https://blueprints.launchpad.net/glance/+spec/glance-bittorrent-delivery

[49] O. Foundation, "Disk and container formats," February 2015. [Online]. Available: http://docs.openstack.org/developer/glance/formats.html

[50] D. Karlson, "Ova's and ovf's: What are they, and what's the difference?" Nov. 2010. [Online]. Available: https://damiankarlson.com/2010/11/01/ovas-and-ovfs-what-are-they-and-whats-the-difference/

[51] "Estobuntu - mugav ja ilus eestikeelne linux." [Online]. Available: http://www.estobuntu.org/

[52] "Livecd customization." [Online]. Available: https://help.ubuntu.com/community/LiveCDCustomization

[53] "What is vanilla?" [Online]. Available: http://whatis.techtarget.com/definition/vanilla

[54] "Ask ubuntu: updates - 14.04 lts apt-get segfault," Oct. 2014. [Online]. Available: http://askubuntu.com/questions/532200/14-04-lts-apt-get-segfault

[55] "Upgrading to libreoffice 4.3," Oct. 2014. [Online]. Available: https://www.linuxquestions.org/questions/showthread.php?s=e0e2f7689f847a56e8cee94a0cafd6bd&p=5216367#post5216367

[56] S. Kemp, "Avoiding slow package updates with package diffs," Sep. 2006. [Online]. Available: https://www.debian-administration.org/article/439/Avoiding_slow_package_updates_with_package_diffs

[57] "Fedora project wiki: Features/presto," May 2009. [Online]. Available: http://fedoraproject.org/wiki/Features/Presto

[58] T. Nakanishi, H.-H. Shih, K. Hisazumi, and A. Fukuda, "A software update scheme by airwaves for automotive equipment," 2013.

[59] "Falcon framework." [Online]. Available: http://falconframework.org/

[60] T. Parkin, "Tools and distributions for embedded linux development," Apr. 2010. [Online]. Available: http://lwn.net/Articles/384713/

[61] L. Poettering, "Revisiting how we put together linux systems," Sep. 2014. [Online]. Available: http://0pointer.net/blog/revisiting-how-we-put-together-linux-systems.html

[62] B. Cohen, “Incentives build robustness in bittorrent,” May 2003. [Online]. Available: bittorrent.org/bittorrentecon.pdf

[63] “Bittorrent: Bep 0003,” 1900. [Online]. Available: http://www.bittorrent.org/beps/bep_0003.html#info-dictionary

[64] A. Norberg, “libtorrent reference: Create torrents.” [Online]. Available: http://www.libtorrent.org/reference-Create_Torrents.html#id5

[65] “Vuze wiki: Torrent piece size,” Mar. 2010. [Online]. Available: https://wiki.vuze.com/mediawiki/index.php?title=Torrent_Piece_Size&oldid=8064

[66] “Bitcomet wiki: Align file to piece boundary,” May 2010. [Online]. Available: http://wiki.bitcomet.com/align_file_to_piece_boundary

[67] “Bittorrent wishlist,” Apr. 2014. [Online]. Available: https://wiki.theory.org/index.php?title=BitTorrentWishList&oldid=2644

[68] “More checksumming algorithms.” [Online]. Available: https://btrfs.wiki.kernel.org/index.php/Project_ideas#More_checksumming_algorithms

[69] “What checksum function does btrfs use?” [Online]. Available: https://btrfs.wiki.kernel.org/index.php/FAQ#What_checksum_function_does_Btrfs_use.3F

[70] “mkfs: change default metadata blocksize to 16kb.” [Online]. Available: https://git.kernel.org/cgit/linux/kernel/git/mason/btrfs-progs.git/commit/?id=c652e4efb8e2dd76ef1627d8cd649c6af5905902

[71] A. Grünbacher, “Posix access control lists on linux,” Apr. 2003. [Online]. Available: http://users.suse.com/~agruen/acl/linux-acls/online/

[72] H. Shirwadkar, “Btrfs content storage mode,” Mar. 2015. [Online]. Available: http://harshadjs.github.io/2015/03/27/Fedora-BTRFS-Content-Storage-Mode/

[73] “Content based storage.” [Online]. Available: https://btrfs.wiki.kernel.org/index.php/Project_ideas#Content_based_storage

[74] E. V. der Sar, “Bittorrent ’s bram cohen patents revolutionary live streaming protocol,” Mar. 2013. [Online]. Available: http://torrentfreak.com/bittorrent-s-bram-cohen-patents-revolutionary-live-streaming-protocol-130326/

[75] A. Jansen, “Index of /pub//scm/linux/kernel/git/arne/far-progs.git,” Sep. 2013. [Online]. Available: https://www.kernel.org/pub//scm/linux/kernel/git/arne/far-progs.git

[76] H. Nordström, “Bug 748071 - grub2 fails to install in the btrfs bootloader area,” Oct. 2011. [Online]. Available: https://bugzilla.redhat.com/show_bug.cgi?id=748071

[77] "Btrfs wiki: Does grub support btrfs." [Online]. Available: https://btrfs.wiki.kernel.org/index.php?title=FAQ&oldid=29348#Does_grub_support_btrfs.3F

[78] "ms-sys." [Online]. Available: http://ms-sys.sourceforge.net/

[79] B. Desmond, "The difference between booting mbr and gpt with grub," Oct. 2012. [Online]. Available: http://www.anchor.com.au/blog/2012/10/the-difference-between-booting-mbr-and-gpt-with-grub/

[80] "The unified efi specification defines an interface between an operating system and platform firmware." [Online]. Available: http://www.intel.com/content/www/us/en/architecture-and-technology/unified-extensible-firmware-interface/efi-specifications-general-technology.html

[81] "Efi boot manager." [Online]. Available: http://linux.dell.com/files/efibootmgr/efibootmgr-0.5.4/efibootmgr.txt

[82] "fstrim - discard unused blocks on a mounted filesystem." [Online]. Available: http://man7.org/linux/man-pages/man8/fstrim.8.html

[83] R. Coker, "3.14.2 debian kernel btrfs corruption after balance," May 2014. [Online]. Available: http://www.spinics.net/lists/linux-btrfs/msg34221.html

[84] "Current stable kernels (< 3.19.4) may fail to mount btrfs and deadlock on boot," Apr. 2015. [Online]. Available: http://www.reddit.com/r/linux/comments/31s042/current_stable_kernels_3194_may_fail_to_mount/

[85] "Vabavaratalgud 2015." [Online]. Available: http://www.teemeara.ee/uudised/meediakajastused/digi-vabavaratalgud-2015

[86] P. Wise, "Debian 8 'jessie' released," Apr. 2015. [Online]. Available: http://lwn.net/Articles/641875/

[87] A. Conrad, "Ubuntu 15.04 (vivid vervet) released," Apr. 2015. [Online]. Available: https://lists.ubuntu.com/archives/ubuntu-announce/2015-April/000195.html

Acronyms

API application programming interface. 9, 18

Btrfs B-tree file system. 1–4, 6, 7, 15–18, 20–23, 25–27, 29, 31, 32, 36, 37

CIFS Common Internet Filesystem. 7

CRC-32C Castagnoli variation of CRC-32. 22

Ext4 Fourth Extended Filesystem. 3, 5, 7

FAI Fully Automated Installation. 3

GPT GUID Partition Table. 28

GRUB GRand Unified Bootloader. 27

HDD hard disk drive. 31

HTTP Hypertext Transport Protocol. 18, 19, 30

LVM Logical Volume Management. 5

LXC Linux Containers. 2, 4, 6, 15–18, 36

MBR Master Boot Record. 27, 28

NFS Network File System. 7

NTFS New Technology File System. 3

OVA Open Virtual Appliance. 9

OVF Open Virtualization Format. 9

P2P peer-to-peer. 21

PXE Preboot eXecution Environment. 13, 16

RAID Redundant Array of Independent Disks. 5

SHA-1 Secure Hash Algorithm. 22

SMB Server Message Block. 7

SoC System on a Chip. 7

SSD solid state disk. 30, 31

SSH Secure Shell. 3, 18, 21, 35

TLS Transport Layer Security. 35

UEFI Unified Extensible Firmware Interface. 28

ZFS Zettabyte Filesystem. 4, 6, 7, 15