Sled: System-Loader for Ephemeral Devices

Lincoln Thurlow, Ryan Goodfellow, Stephen Schwab
USC Information Sciences Institute
lincoln@isi.edu, rgoodfel@isi.edu, schwab@isi.edu

***Abstract*—Imaging an operating system onto a server is an extensive and time consuming process, which commonly taking several minutes to boot. For a testbed administrator, loading an image onto a device is one of the slowest yet most relied upon tasks that a testbed must complete prior to starting an experiment. To minimize wait times for disk loading, as well as to streamline the process of image loading, we introduce *Sled*, a system-loader for ephemeral devices, that uses warm-booting to quickly image devices. Sled is able to provision a new operating system in under a minute, and is 10 times faster than previous methods of image loading.**

I. Introduction

In a testbed, experiments are allocated a set of devices. These devices live for the duration of the experiment, once the experiment is finished, the devices are wiped so a new experiment can begin. This life cycle is common across many domains, but especially for testbeds where devices are ephemeral for the duration of an experiment. Experiments are composed of many individual tasks such as assigning devices to an experiment, building network topology, or configuring devices. For testbed administrators, the longest task in preparing an experiment is imaging the devices, it is not uncommon for this process to take tens of minutes to complete, potentially frustrating experimenters in the process.

In order for experiments to be reproducible, testbeds require a clean environment. Typically, this is achieved by wiping the device of the previous state and power cycling **before and after** writing an image to the disk. Many testbeds commonly implement disk imaging using the Frisbee disk imaging protocol [1]. Frisbee aggressively minimizes the number of bytes transmitted across the network at the cost of latency, which we discuss further in Section II.

To reduce the overall time it takes to image a device from scratch for an experiment, we developed *Sled*, an image loading protocol for rapidly loading an image onto a device. Sled incorporates two primary features to improve the performance of disk imaging. First, it utilizes u-root [2], a small golang initramfs (root file system), which can be loaded at minimal cost. Second, from the u-root kernel, Sled is able to kernel execute (kexec) [3] into another kernel, bypassing firmware and hardware initialization (warm boot). Kexec is capable of targeting any kernel with kexec enabled, allowing Sled to target bootloaders such as GNU's GRUB[4] to provide targets for virtually any operating system.

II. Previous Work

As a part of the Emulab testbed, the Emulab team developed Frisbee [1]. Frisbee introduced a custom format able to exploit the operating system's unallocated disk blocks to minimize the number bytes it transmits over the network. Due to the custom image format, the burden for creating new images lies with testbed administrators. Frisbee also uses multicast to reduce the number redundant bytes sent over the network. This introduces latency in the multicast group waiting for new members to join in addition to packets that must be retransmitted to the multicast group. Once the image has been written the the disk, the device reboots, loading the newly written image. Most of the drawbacks outlined above for causing additional latency have been noted by the Emulab team [5].

III. Sled

Sled is designed to be quick. Devices using Sled are capable of booting on the order of seconds rather than minutes. Sled consists of three components: clients (*Sledc*), daemons (*Sledd*), and the controller (*Sledctl*). The controller is responsible for configuring the daemon's datastore, which stores a map between mac addresses and Sled commands. Clients request and execute commands from the daemons. The current Sled commands are *Wipe*, *Write*, and *Kexec*. Between them they are responsible for cleaning the device, writing a disk image, and booting into the new image. Clients run the Sled binary on top of u-root, and are responsible for conforming the device to the experimenter's requirements through the Sled commands.

Sled separates disk wiping from writing. This detail allows for the sanitizing of devices to be amortized on experiment tear-down rather than construction. Disk wipe is implemented by zeroing the disk. Both disk wipe and write are designed to accommodate the heterogeneity of a device, with customizable write buffers to fully saturate IO bandwidth of the device's storage substrate. The last Sled command, kexec, takes the parameters given to it and jumps into the new kernel. After kexecing, the device has completed the image loading process and can be handed over to the experimenter, reducing the need to power cycle after writing the disk image.

A. Sledc

Sledc is written on top of u-root, a minimal initramfs that each experiment device boots from preboot execution environment (PXE) over the network. The total size of the u-root binary is between 10-15MB, depending on supplemental u-root commands. Sledc begins by negotiating an IP with a DHCP server, after which it requests Sled commands from Sledd. If the device has just finished an experiment, Sledd will request Sledc to wipe the disk, preparing it for the next

978-1-7281-1878-9/19/$31.00 ©2019 IEEE

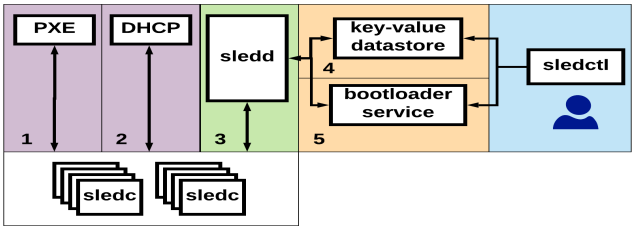

Fig. 1. Device PXE boots u-root image with Sledc (1). Sledc requests an IP (2). Sledc requests Sled commands (3). Sledd looks up Sledc mac in datastore (4). Optionally, Sledd requests bootloader binary (5). Response is returned to client in (2). Sledc executes Sled commands.

experiment. Otherwise, Sledc will receive commands for either writing a new bootable image or kexecing from the u-root kernel to the kernel specified in the Sled command. Once the boot has completed, control is given over to the experimenter. The overview of Sled communication is shown in Figure 1.

B. Sledd

Sledd is a stateless service intended to be deployed in a distributed manner. Sledd listens on two ports, one for Sled command requests, and the other for transferring disk images. State is stored in a distributed datastore which each daemon connects in order to service requests. The limitations to Sledd's scalability is in the number of write requests multiplied by the write request's buffer size. Additional work is planned for batching requests and auto-scaling buffers to provide better performance and fairness for heterogeneous devices.

C. Sledctl

Sledctl supports both an API and command line interface to the Sled control plane. The Sled control plane is responsible for mapping mac addresses of devices with bootable images in the distributed datastore accessed by Sledd. Through the Sledctl, testbed administrators manage what images are loaded onto what devices as defined by experimenters.

IV. Results

We present preliminary results testing Sled against Frisbee using the DETERLab testbed [6]. We used two HP Proliant DL360 G8 Servers [7] with a 1Gb link used for copying images, mimicking the Frisbee implementation in DETERLab. To test the performance of Frisbee, we used the Emulab command *os_load* to request Frisbee write an image to the server. The two measurements in Figure 2 are: how long *os_load* took to write and prepare the disk image (Disk Written), and the total amount of time until the new image presented a prompt to the experimenter (Operational). For Sled, we measured the amount of time it took to write the operating system from one HP G8 Server acting as Sledd to the other HP G8 server as Sledc. The measurements are far from perfect, as the single server scenario for Frisbee unnecessarily waits for other multicast consumers to join the group. Additionally, Frisbee used Deterlab's shared control network while Sled used the dedicated experiment network.

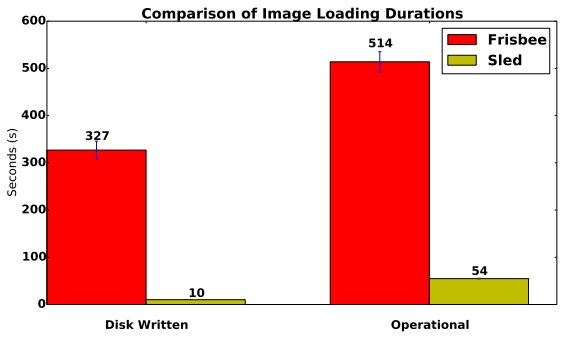

Fig. 2. The latency of writing and loading an Ubuntu16.04 onto a HP Proliant DL360 server using Frisbee and Sled.

Each test was run five times; these results indicate that Sled is roughly one-tenth the latency, or ten times faster than Frisbee for booting an image.

V. Conclusion

Sled is a sub-minute image bootloader developed for testbed administrators to reduce the time, complexity, and overhead necessary to load an operating system onto devices in a testbed. Sled has no disk format requirements and allows users to generate and supply their own images to testbeds. Sled is approximately 10 times faster than Frisbee for imaging devices. For future work we plan on deploying Sled to our cluster of over 1400 nodes.

Acknowledgments

This material is based upon work supported by Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001117C0053. The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

Availability

Sled is currently available on gitlab under an Apache 2.0 license.

https://gitlab.com/mergetb/tech/sled

References

[1] M. Hibler, L. Stoller, J. Lepreau, R. Ricci, and C. Barb, "Fast, Scalable Disk Imaging with Frisbee," in *USENIX Annual Technical Conference, General Track*, 2003, pp. 283–296.
[2] R. G. Minnich and A. Mirtchovski, "U-root: A Go-based, Firmware Embeddable Root File System with On-demand Compilation." in *USENIX Annual Technical Conference*, 2015, pp. 577–586.
[3] J. Corbet, "Kexec." [Online]. Available: https://lwn.net/Articles/15468/
[4] B. Dubbs, "GNU GRUB." [Online]. Available: https://www.gnu.org/software/grub/
[5] L. Stoller, "Potential areas of improvement for a new Frisbee Hackfest." [Online]. Available: https://gitlab.flux.utah.edu/emulab/emulab-devel/blob/master/clientside/os/frisbee.redux/IDEAS
[6] T. Benzel, "The Science of Cyber Security Experimentation: The DETER Project," in *Proceedings of the 27th Annual Computer Security Applications Conference*. ACM, 2011, pp. 137–148.
[7] DETERLab, "Node types," https://docs.deterlab.net/core/node-types/#dl380g3, 2018.