systemd
our next-generation init system

PABLO N. HESS
Instructor
Red Hat São Paulo
December 2011

General info & History

Authors:

Lennart Poettering (Red Hat)
Pulseaudio, Avahi

Kay Sievers (openSUSE)
Udev

Spelling:

It's systemd

not system D
not System D
not SystemD
not system d

Current default init for:

Fedora

openSUSE

Mandriva

Future default init for:

Gentoo

Arch

Mageia

Probably everyone else

Major features

Massively parallel service initialization

Replaces Upstart and SysVinit

On-demand network service initialization

Replaces (x)inetd

On-demand fsck'ing & mounting

Replaces fstab and autofs

On-demand socket-based initialization

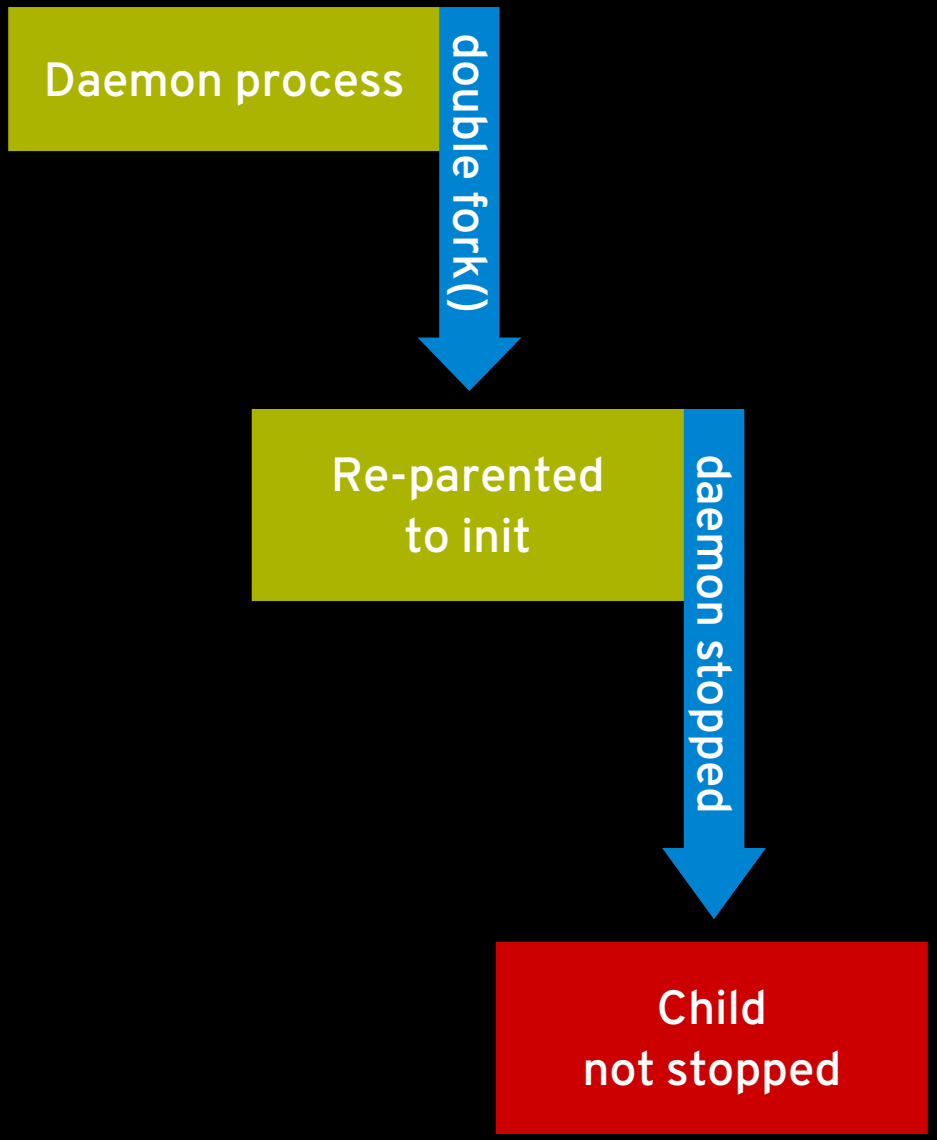
Better than Upstart

Motivation:

What current init systems do not/can not provide

- Reliable supervisioning
- Reliable dependencies
- Parallel service initialization
- Socket-based initialization
- Better-than-shellscript speeds
- Code deduplication/ sanitization
- Low first user PIDs
- Path-based initialization

SysV/Upstart



Reliable supervisioning

Reliable dependencies

Parallel service initialization

Socket-based initialization

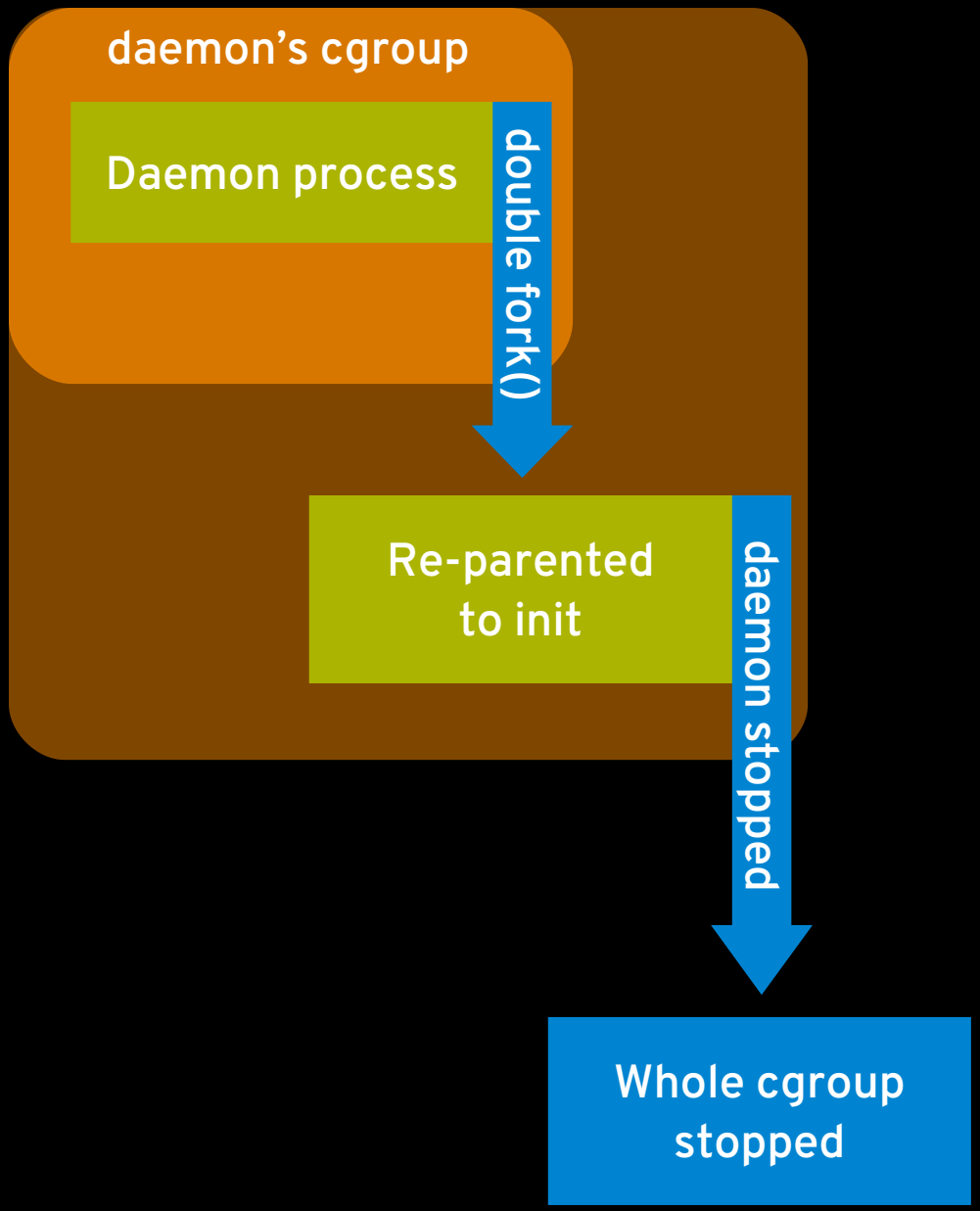
Better-than-shellscript speeds

Code deduplication/ sanitization

Low first user PIDs

Path-based initialization

systemd



Reliable supervisioning

Reliable dependencies

Parallel service initialization

Socket-based initialization

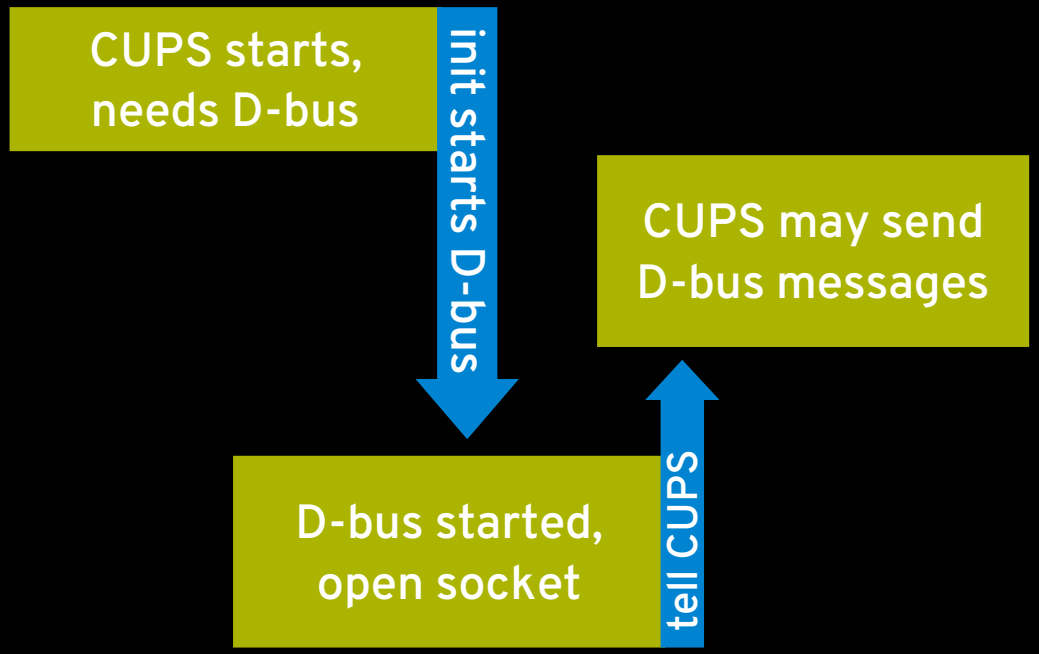
Better-than-shellscript speeds

Code deduplication/ sanitization

Low first user PIDs

Path-based initialization

SysV/Upstart



Reliable supervisioning

Reliable dependencies

Parallel service initialization

Socket-based initialization

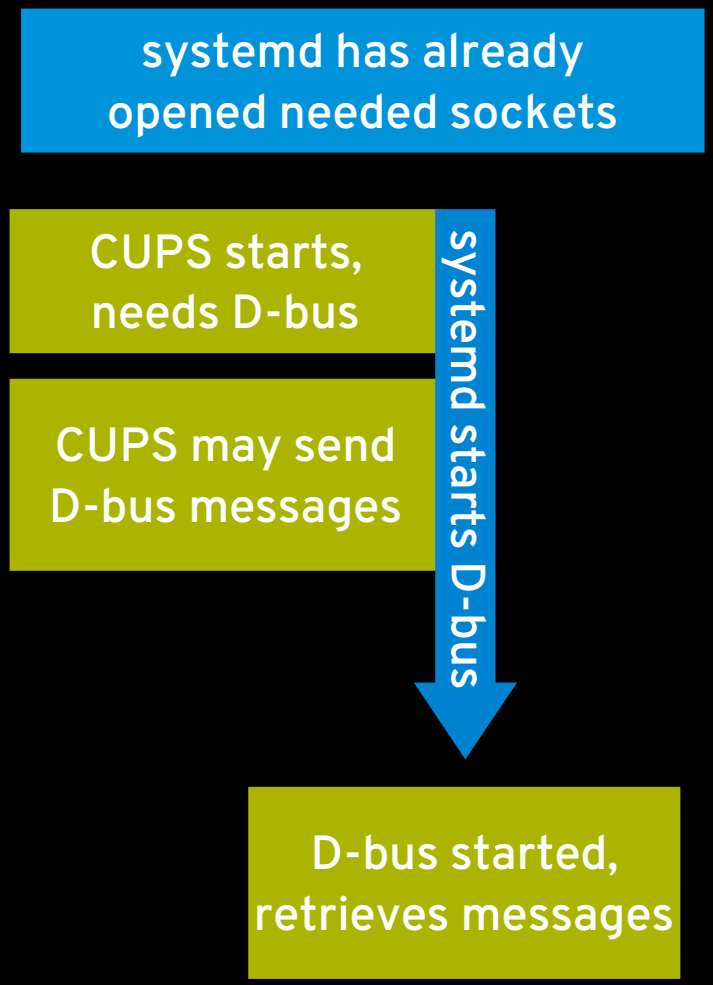
Better-than-shellscript speeds

Code deduplication/ sanitization

Low first user PIDs

Path-based initialization

systemd



Reliable supervisioning

Reliable dependencies

Parallel service initialization

Socket-based initialization

Better-than-shellscript speeds

Code deduplication/ sanitization

Low first user PIDs

Path-based initialization

SystemV

```
start() {
 [ -x $exec ] || exit 5
 # Source config
 if [ -f /etc/sysconfig/rsyslog ] ;
   then
   . /etc/sysconfig/rsyslog
 fi
 umask 077
 echo -n $"Starting system logger: "
 daemon --pidfile="${PIDFILE}" \
   $exec $SYSLOGD_OPTIONS
 RETVAL=$?
 echo
 [ $RETVAL -eq 0 ] && touch $lockfile
 return $RETVAL
}
```

```
ExecStartPre=/bin/systemctl stop \
  systemd-kmsg-syslogd.service
ExecStart=/usr/sbin/rsyslogd -n -c5
Sockets=syslog.socket
StandardOutput=null
```

systemd

- Reliable supervisioning
- Reliable dependencies
- Parallel service initialization
- Socket-based initialization
- Better-than-shellscript speeds
- Code deduplication/ sanitization
- Low first user PIDs
- Path-based initialization

Rsyslog as an example

sysvinit script

- 106 lines
- 75 lines of code

systemd “unit file”

- 11 lines
- 9 lines of code

Reliable supervisioning

Reliable dependencies

Parallel service initialization

Socket-based initialization

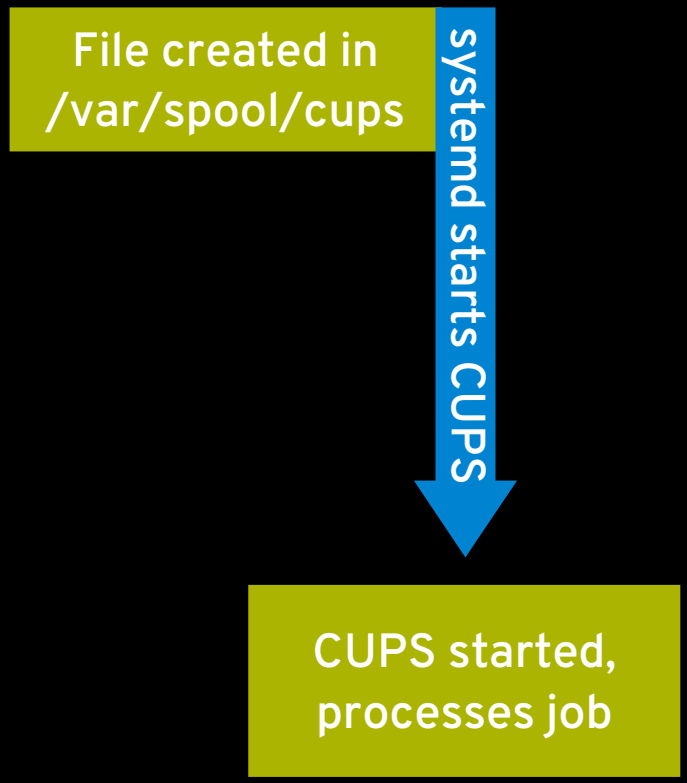
Better-than-shellscript speeds

Code deduplication/ sanitization

Low first user PIDs

Path-based initialization

CUPS as an example



Reliable supervisioning

Reliable dependencies

Parallel service initialization

Socket-based initialization

Better-than-shellscript speeds

Code deduplication/ sanitization

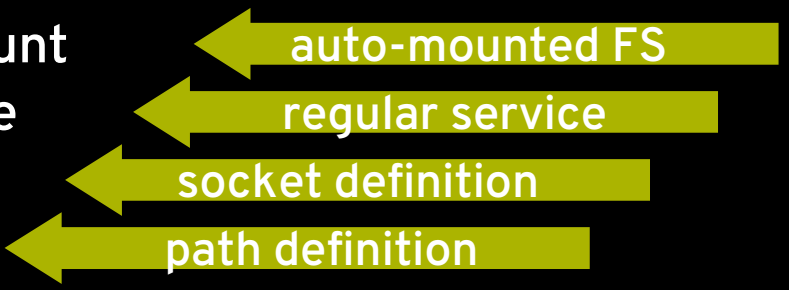
Low first user PIDs

Path-based initialization

Working with systemd

Everything is a unit:

- home.automount
- rsyslog.service
- sshd.socket
- cups.path



Targets “want” units

- multi-user.target.wants/
  - postfix.service
  - cron.target
  - sysinit.target
- sysinit.target.wants/
  - remount-rootfs.service
  - quotaon.service

Invocation: systemd versus SystemV

SystemV	systemd
`# service sshd start`	`# systemctl start sshd.service`
`# chkconfig sshd on`	`# systemctl enable sshd.service`
add autofs map	`# systemctl enable home.automount`
add fstab entry	`# systemctl enable home.mount`
`# init 5`	`# systemctl isolate graphical.target`

Unit files example: automount & mount

home.automount

```
[Unit]
Description=Automount my /home

[Automount]
Where=/home

[Install]
WantedBy=sysinit.target
```

man systemd.automount

home.mount

```
[Unit]
Description=My home directory

[Mount]
#What=UUID=fd6e2ed9-d430-45b3-9...
What=/dev/sdb9
Where=/home
Type=ext4
Options=noatime,discard,nobarrier
```

man systemd.mount

Unit files example: swap

dev-sda5.swap

```
[Unit]
Description=Swap on /dev/sda5

[Swap]
What=/dev/sda5
Priority=1
TimeoutSec=5

[Install]
WantedBy=swap.target
```

man systemd.swap

Unit files example: services

sshd.service

```
[Unit]
Description=OpenSSH server daemon.
After=syslog.target network.target auditd.service

[Service]
Type=simple
ExecStart=/usr/sbin/sshd -D
ExecReload=/bin/kill -HUP $MAINPID

[Install]
WantedBy=multi-user.target
```

or

man systemd.service

Unit files example: services

getty@.service

```
...
[Service]
Environment=TERM=linux
ExecStart=-/sbin/agetty %I 38400
Restart=always
RestartSec=0
UtmpIdentifier=%I
TTYPath=/dev/%I
...
```

or

```
# systemctl --full --no-pager |grep getty

getty@tty2.service
getty@tty3.service
getty@tty4.service
getty@tty5.service
getty@tty6.service
```

man systemd.service

Unit files example: services

sshd@.service

```
[Unit]
Description=SSH Per-Connection
Server
After=syslog.target

[Service]
ExecStart=/usr/sbin/sshd -i
StandardInput=socket
```

+

sshd.socket

```
[Unit]
Conflicts=sshd.service

[Socket]
ListenStream=22
ListenStream=2200
Accept=yes

[Install]
WantedBy=sockets.target
```

man systemd.service

man systemd.socket

```
# systemctl --full --no-pager |grep sshd

sshd@192.168.123.241:22-192.168.123.100:50083.service
sshd@192.168.123.241:22-192.168.123.245:35623.service
sshd@192.168.123.241:22-192.168.123.245:35624.service
sshd@192.168.123.241:22-192.168.123.245:60016.service
sshd@192.168.123.241:2200-66.187.233.202:11574.service
```

Unit files example: services (oneshot)

iptables.service

```
[Unit]
Description=IPv4 firewall with iptables
After=syslog.target
ConditionPathExists=/etc/sysconfig/iptables

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/usr/libexec/iptables.init start
ExecStop=/usr/libexec/iptables.init stop
Environment=BOOTUP=serial
Environment=CONSOLETYPE=serial
StandardOutput=syslog
StandardError=syslog

[Install]
WantedBy=basic.target
```

Unit files example: services (forking)

dnsmasq.service

```
[Unit]
Description=DNS caching server.
After=syslog.target network.target

[Service]
Type=forking
PIDFile=/var/run/dnsmasq.pid
EnvironmentFile=-/etc/sysconfig/network
ExecStart=/usr/sbin/dnsmasq -s $HOSTNAME

[Install]
WantedBy=multi-user.target
```

the choice for legacy SysV init scripts

One dir for the packager

```
$ ls /lib/systemd/system
abrt-ccpp.service                          poweroff.service
abrtd.service                              poweroff.target
abrt-oops.service                          poweroff.target.wants
abrt-vmcore.service                        pppoe-server.service
accounts-daemon.service                    prefdm.service
alsa-restore.service                       printer.target
alsa-store.service                         proc-sys-fs-binfmt_misc.automount
anaconda-shell@.service                    proc-sys-fs-binfmt_misc.mount
anaconda.target                            psacct.service
arp-ethers.service                         quotacheck.service
atd.service                                quotaon.service
auditd.service                             rc-local.service
autovt@.service                            rdisc.service
avahi-daemon.service                       reboot.service
avahi-daemon.socket                        reboot.target
basic.target                               reboot.target.wants
basic.target.wants                         remote-fs.target
bluetooth.service                          remount-rootfs.service
bluetooth.target                           rescue.service
canberra-system-bootup.service             rescue.target
canberra-system-shutdown-reboot.service    restorecond.service
canberra-system-shutdown.service           rpcbind.target
chronyd.service                            rsyslog.service
chrony-wait.service                        rtkit-daemon.service
```

One dir for the packager

...and one for the sysadmin

```
$ ls /lib/systemd/system
abr
abr
abr
abr
acc
als
als
ana
ana
arp
atd
aud
aut
ava
ava
bas
bas
blu
blu
can
can
can
chr
chr
```

```
$ ls /etc/systemd/system
home.automount                                  home.mount
basic.target.wants                              graphical.target.wants
bluetooth.target.wants                          multi-user.target.wants
my-own-target.target.wants                      network.target.wants
dbus-org.freedesktop.NetworkManager.service     printer.target.wants
default.target                                  sockets.target.wants
default.target.wants                            sysinit.target.wants
getty.target.wants
```

Troubleshooting

Select a target (“runlevel”) at boot time:

```
kernel /vmlinuz-3.1 (...) systemd.target=emergency.target
```

loads the basic stuff

```
kernel /vmlinuz-3.1 (...) systemd.target=multi-user.target
```

equivalent to runlevel 3

```
kernel /vmlinuz-3.1 (...) systemd.log_level=debug
```

sets log level

```
kernel /vmlinuz-3.1 (...) systemd.log_target=kmsg
```

logs to dmesg

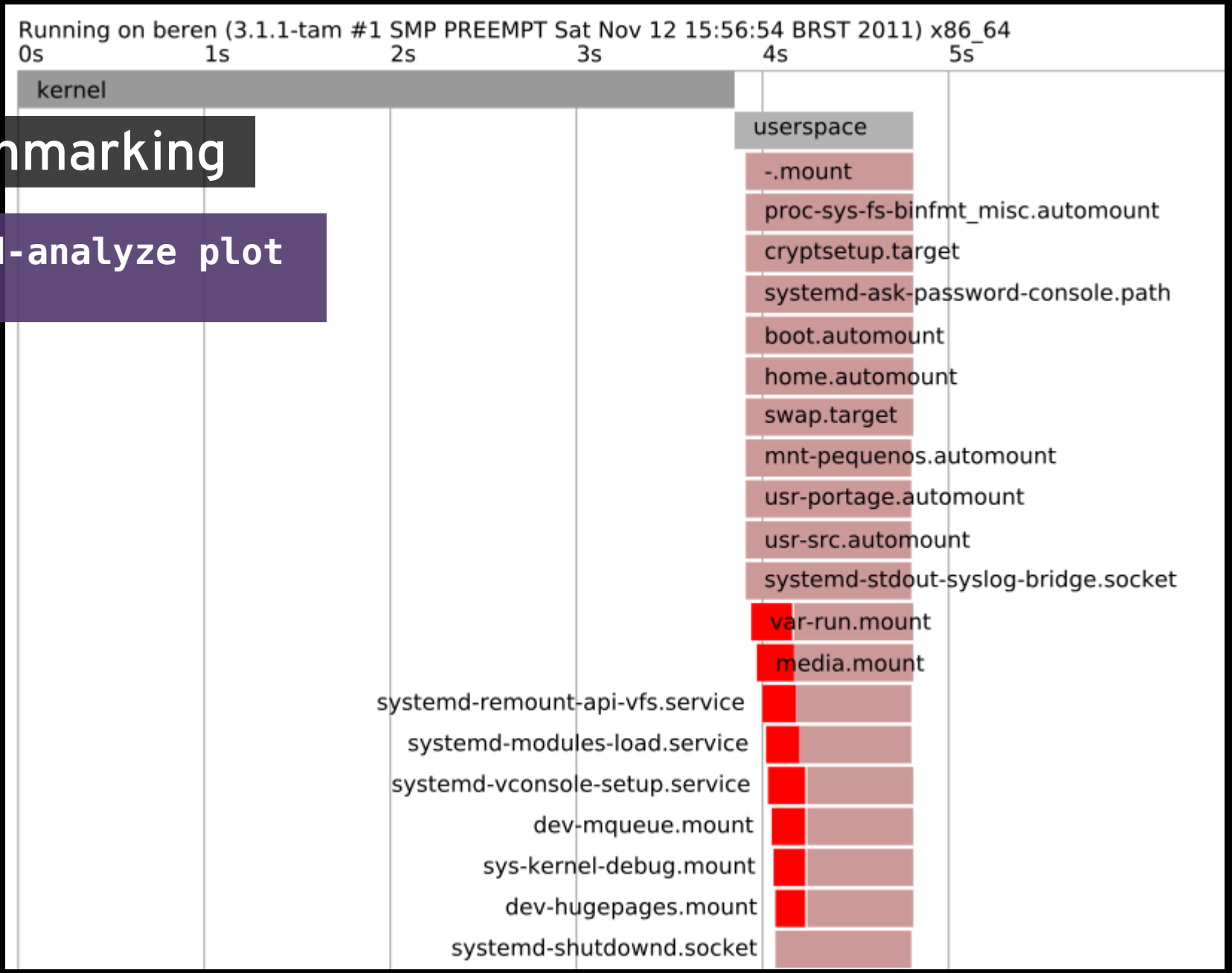
Benchmarking

```
# systemd-analyze time

Startup finished in 1812ms (kernel) + 3722ms (initramfs) +
3912ms (userspace) = 9446ms
```

```
# systemd-analyze blame

9682ms sshd-keygen.service
4483ms abrtd.service
4382ms plymouth-start.service
4365ms systemd-readahead-replay.service
2268ms sendmail.service
2182ms udev-settle.service
 ...
  16ms rpcbind.service
  13ms dnsmasq.service
```

Benchmarking

```
# systemd-analyze plot
```



PABLO N. HESS
Instructor
Red Hat São Paulo
December 2011