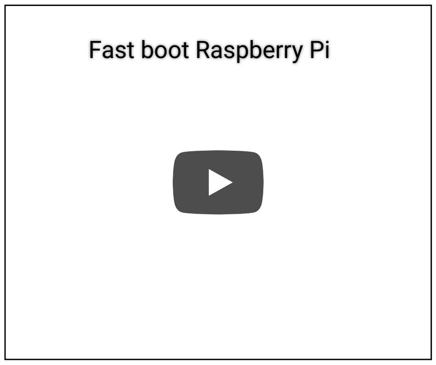
Himesh's Blog

Stuff you may find interesting

Wednesday, August 1, 2018

Fast boot with Raspberry Pi

I am hoping to have a raspberry pi power a wildlife camera. This camera will have to rely on battery and solar power. As a result, it would be beneficial if the camera was off when no wildlife is present. To aid in this regard, I hope to use a motion sensor that can trigger the raspberry pi to turn on and take a picture. For this to work, the time from motion detection to picture snap is heavily influenced by the boot time of the raspberry pi. Here is a video of what I've been able to accomplish:



I am starting with the stock Raspbian Stretch Lite distribution on a Pi 3B. Boot times out of the box are on the order of 1 minute. Boot time is influenced by the following:

```
1. Hardware
2. Bootloader
3. Kernel
4. Userspace
```

The Raspberry Pi hardware and bootloader are essentially out of my control. There was an effort to open source the boot loader, however the proprietary binary blob is the only reasonable option at this point. The Hardware and bootloader take approximately a minimum of **1.5-2 seconds** to run. This is explained in an excellent post on the Raspberry Pi Forums. The author tested boot times with various minimal boot loaders. The fastest any code could be run on the ARM processor was around **1.5 seconds**.

I was able to get the **kernel** and **userspace** boot times down to about **0.6 second** and **0.8 seconds** respectively. As a result my **total boot time** is on the order of **3.5 to 4 seconds** (from power on to picture taken).

To be able to control the Raspberry Pi without SSH, I used serial (UART) communications. See my previous post to learn how.

I reduced the kernel and userspace boot times by doing the following (in order highest yield to lowest yield):

1. Editing the `/boot/config.txt` with the following changes:

```
# Disable the rainbow splash screen
disable_splash=1

# Disable bluetooth
dtoverlay=pi3-disable-bt

#Disable Wifi
dtoverlay=pi3-disable-wifi

# Overclock the SD Card from 50 to 100MHz
# This can only be done with at least a UHS Class 1 card
dtoverlay=sdtweak,overclock_50=100

# Set the bootloader delay to 0 seconds. The default is 1s if not specified.
boot_delay=0

# Overclock the raspberry pi. This voids its warranty. Make sure you have a good power supply.
```

Popular Posts

Fast boot
I am hopin
raspberry
wildlife ca
camera wi
battery an
a result, it would ...

Raspberry
Issues - Ult
Update: Th
will probab
warranty a
guarantees
own risk. I have been stru

Wireless, E
Time-lapse
Raspberry
I was looki
projects fo
Pi and cam
cool project by Jeremy Bl
powered, Wireless, Motio

Small and
Adapter fo
Pi
I wanted t
the benefi
owners wh
for an appropriate and ec
adapter. The che...

First Post
This is a content-less first
I can figure out what my b
like.

Raspberry
Module HD
So a bunch
modules w
available l
finally wer
ordered one for my Raspb
in le...

Serial Com
Raspberry
Running a
raspberry
challengin
been using
my raspberry pi. This wor
ra...

PiJuice - M
the Raspbe
Impression
Ever since
came out I
make a tin
that would take pictures
months, or even a...

Raspberry Pi Camera Mo
I am very excited about
Camera module. So excite
compiled the latest infor
find on the camera ...

Resizing partitions within
I wanted to backup a 32G
16GB SD Card. The 32 GB
contained 10GB of data so
possible. I started by ...

```
force_turbo=1
```

2. Make the kernel output less verbose by adding the "quiet" flag to the kernel command line in file /boot/cmdline.txt

```
dwc_otg.lpm_enable=0 console=serial0,115200 console=tty1 root=PARTUUID=32e07f87-02 rootfstype=ext4
elevator=deadline fsck.repair=yes quiet rootwait
```

3. Use systemd-analyze blame, systemd-analyze critical-chain to disable services I didn't need

```
sudo systemctl disable dhcpcd.service
sudo systemctl disable networking.service
sudo systemctl disable ssh.service
sudo systemctl disable ntp.service
sudo systemctl disable dphys-swapfile.service
sudo systemctl disable keyboard-setup.service
sudo systemctl disable apt-daily.service
sudo systemctl disable wifi-country.service
sudo systemctl disable hciuart.service
sudo systemctl disable raspi-config.service
sudo systemctl disable avahi-daemon.service
sudo systemctl disable triggerhappy.service
```

See the references below to learn about a primer on systemd and the new linux init system to learn about how to interpret and write the above services.

4. Add a service that runs the code you would like to run as fast as possible. For example if you wanted to add a service called "1ylapse", create the following file: `/etc/systemd/system/1ylapse.service`

```
[Unit]
Description=Starts 1 Year Lapse Service

[Service]
ExecStart=/home/pi/foo.sh
StandardOutput=syslog
StandardError=syslog
SyslogIdentifier=piservice
User=pi
Group=pi
WorkingDirectory=/root/1ylapse/

[Install]
WantedBy=basic.target
```

5. Analyze the kernel for unnecessary work being done at boot.

To do this you need to compile your kernel with "CONFIG_PRINTK_TIME" and "CONFIG_KALLSYMS". This should be enabled on the default raspberry pi kernel. This allows you to add "initcall_debug" to the kernel command line. The kernel will now output start and end time information for every init call. You can use "bootgraph.pl" which is included with the linux kernel to analyze the output of dmesg.

On the raspberry pi:

```
$ dmesg > boot.log
```

On the cross-compile host:

```
$ linux/scripts/bootgraph.pl boot.log > boot.sv
```

This will output an graph of what is taking the most time when initializing the kernel. I noticed that a routine used by the USB driver was taking around 0.3s. I don't need USB for my project so I disabled USB support when re-compiling the kernel (see below). This saved around **0.3s**.

6. Re-compile the Linux kernel

Remove stuff that is wasting time during initialization. I used the guide from the Raspberry Pi Foundation to learn how to re-compile the kernel.

7. Use LZO compression for kernel

When compiling the Linux kernel, select "LZO" compression instead of "GZip". This saved around **0.3s**.

8. Don't re-mount the /boot partition

Edit the **/etc/fstab** file and comment out the line that re-mounts the /boot partition. This saved around **0.2s**.

The final systemd-analyze shows:

```
Startup finished in 669ms (kernel) + 1.225s (userspace) = 1.894s
```

It should be noted that my camera service starts before systemd is finished initializing. You can find out when your service starts by using **systemd-analyze crritical-chain**. You can see below that my service starts at 836ms after the kernel is finished initializing, rather than the total of 1.225s.

```
$ systemd-analyze critical-chain 1ylapse.service
1ylapse.service @836ms
└─basic.target @832ms
  └─sockets.target @832ms
    └─dbus.socket @831ms
      └─sysinit.target @826ms
        └─systemd-update-utmp.service @784ms +41ms
          └─systemd-tmpfiles-setup.service @748ms +33ms
            └─systemd-journal-flush.service @658ms +87ms
              └─systemd-remount-fs.service @585ms +64ms
                └─systemd-fsck-root.service @444ms +137ms
                  └─systemd-journald.socket @433ms
                    └─-.slice @376ms
```

9. Remove plymouth to disable systemd init messages

```
sudo apt-get purge --remove plymouth
```

I haven't seen anyone boot a raspberry pi faster than this using full Raspbian. Bare metal is obviously faster however. However having full Raspbian available at this boot up speed is a good compromise.

Things that failed to improve boot time included making the root partition read only.

Hopefully this helps others in my predicament.

References:

1. Presentation by Jan Altenberg on booting linux in less than 1 second. Powerpoint here. Youtube of presentation here.
2. Excellent powerpoint on boot time optimization using a beagle bone as a prototype here.
3. Excellent powerpoint on speeding up raspberry pi boot time here.
4. Excellent primer on systemd-anzlyze.
5. Good stackoverflow question on using sytemd-analyze.

Posted by Himesh Prasad at 2:10 PM

8 comments:

Ash McKenzie August 12, 2018 at 4:28 PM

This is incredible, thankyou!

Reply

Artur Rodak August 13, 2018 at 11:37 PM

Great tutorial. Thank you.

Reply

Unknown January 16, 2019 at 10:03 AM

Wow!

Reply

Murat Demir March 19, 2019 at 5:13 AM

Amazing. Do you think an Rpi zero can reach the similar boot time?

Reply

NonTechGuy April 13, 2019 at 10:53 PM

Awesome tutorial, maybe you should male a longer and more detailed video on this, kudos!!

Reply

Unknown May 20, 2019 at 7:53 AM

Very interesting. What camera software are you using?

Reply

ukdutypaid September 5, 2019 at 6:56 AM

The most recent comments here have been spam so I am unfollowing what could have been interesting followup.

Reply

JimiHx April 8, 2020 at 1:08 PM

"sudo apt-get purge --remove plymouth" will leave your raspbian unbootable because there is a depency to mountall which is removed when executing the above command.

Reply

Enter your comment...

Commenting as thomaschristian@gmail Comment as: Tom Christian (Sign out

Publish Preview Notify me

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

Simple theme. Powered by Blogger.